504-REN-96/004
CSC10031821

MISSION OPERATIONS AND DATA SYSTEMS DIRECTORATE

Renaissance Generic Architecture

Version 2.0

Volume 2 of 2
Appendices

May 1996

National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland

Renaissance Generic Architecture

Version 2.0

Volume 2 0f 2
Appendices

May 1996

Prepared Under Contract NAS5-31000
Task Assignment 05-601

Approved By:

Gary F. Meyers Date
Systems Engineering Office, Code 504

Goddard Space Flight Center
Greenbelt, Maryland

Preface

This document contains the appendices accompanying the architecture prepared by the Renaissance Team for the second generation aproach for Mission Operations and Data Systems Directorate (MO&DSD)-developed ground data systems. The architecture description is released as a separate volume, 504-REN-96/003. These appendices contain supplementary material for the architecture definition. The material includes requirements and other sources driving the architecture, examples of mission systems based on the architecture, and approaches within the architecture to technical performance.

This design effort started during Fiscal Year (FY) 1995 and was completed in FY96, under the auspices of the Renaissance program in Code 504.

This document is under the configuration management of the Systems Engineering Office.

Configuration Change Requests (CCRs) to this document shall be submitted to the Systems Engineering Office, along with supportive material justifying the proposed change. Changes to this document shall be made by document change notice (DCN) or by complete revision.

Questions and proposed changes concerning this document shall be addressed to:

Gary F. Meyers, Systems Engineering Office
Code 504
Goddard Space Flight Center
Greenbelt, Maryland 20771

Contents

Appendix A. Second Generation Concepts

Appendix B. Second Generation Mission Requirements

Appendix C. Architecture Alternatives and Analysis

Appendix D. Capabilities and Design Drivers

Appendix E. University Mission Design

Appendix F. ASIST Oriented Design

Appendix G. NASA-Integrated Mission Design

Appendix H. Technical Performance Approaches

Abbreviations and Acronyms

Glossary

List of Figures

List of Tables

Appendix A. Second Generation Concepts

A.1 Purpose

This document presents a preliminary approach to identifying the issues and operational considerations that may apply to second generation Renaissance systems and which must be addressed by the architecture for these. The document includes a summary mission life-cycle model, which reflects the impacts of some of the anticipated changes in the development process. The second generation operations concepts are described in the third section. Finally, an approach to assessing the cost impacts of these concepts is discussed in the fourth section.

A.2 Mission Life-Cycle Model

This section presents a model of the mission life cycle for a second generation Renaissance mission. The model describes the anticipated activities and the potential impacts for a Renaissance approach to the activities. The life-cycle model consists of three phases, with the compression reflecting both the need and implementations for shorter development times.

A.2.1 Analysis Phase

The analysis phase of a mission is more properly that devoted to selecting missions. Generally, possible mission characteristics are identified and weighed for value, cost and feasibility. The results of this process are a selected mission, or missions, along with mission descriptions in the form of an operations concept, preliminary design, preliminary cost and budget, and schedule.

The anticipated impact of the Renaissance approach to this phase arises from the expected presence of a repository of approaches to implementation and operations that can be implemented using Renaissance off-the-shelf building blocks. The expectation is that this repository can be used in mission analysis to assess implementation feasibility and cost with a low labor impact and rapid turn-around. By using this, not only is the analysis period shortened, but the resulting operations concepts, design and cost estimates are both more accurate and usable in the implementation phase. If a Mission Operations Control Architecture (MOCA)-like approach is used, this would be very helpful in assessing space-ground implementation trades. An option, even for this phase, is to use the repository to implement prototypes for feasibility assessment.

Table A-1 details the expected activities during this phase. It should be noted that mission assessment is an iterative process, and essentially the same activity occurs during selection of the mission from the alternatives.

Table A-1. Mission Analysis Phase Activities

Activity	Input/Output
Mission goals collection	Input from scientists Output: statement of mission goals, objectives
Mission goals evaluation, done by peer review	Input: Mission goals alternatives Output: Priority order goals, evaluation
Mission alternatives evaluation of feasibility and costs	Input: Goals, implementation approaches, cost models. Output: Feasibility and cost assessments for alternative missions
Mission selection	Input: Alternative mission goals, feasibility and cost assessments Output: Selection of mission
Mission concept development: develop operations concepts and designs	Input: Goals, implementation approaches, cost models Output: Feasibility and cost assessments for mission Preliminary operations concepts and designs

A.2.2 Implementation Phase

Table A-2 details the expected activities during the implementation phase.

Table A-2. Mission Implementation Phase Activities

Activity	Input/Output
Mission S/C implementation (includes S/C I&T)	Input: Preliminary design, specifications for interfaces Output: spacecraft
Mission instrument development	Input: Mission goals, preliminary design, spacecraft specification Output: Priority order goals, evaluation
Mission Ground Data System implementation	Input: Goals, implementation approaches, preliminary design. Output: GDS system with customization
Mission integration and test	Input: Alternative mission goals, feasibility and cost assessments Output: Selection of mission
Mission training	Input: Designs, operations procedures, preliminary versions Output: Staff training and operations documentation

A.2.3 Operations Phase

Table A-3 details the expected activities during the operations phase.

Table A-3. Mission Operations Phase Activities

Activity	Input/Output
Mission launch and early orbit operations	
Mission customer data operations	Input: Procedure specifications, customer plans Output: Status and acquired data
Mission infrastructure operations	Input: Plans, operational requirements, status Output: operational infrastructure, customer support operations, status
Mission science operations	Input: Science goals and objectives, Science intermediate results Output: Mission science assessments and plans
Mission maintenance	Input: Trends and problems with mission infrastructure; consumable resources; upgrades and repaired elements Output: Revisions to the operational elements and projections of operational status

A.3 Operations Concepts for Second Generation

The following sections contain brief descriptions of the major innovations anticipated for second generation Renaissance missions. Each section describes one major topic. It is expected that several of these may be implemented within any given mission, in varying combinations. An effort has been made to include all items identified by any known NASA action team as well as those from other customer areas. In many ways this list makes up the "what if" list of items requiring impact assessment.

A.3.1 Integrated Spacecraft

The concept of the integrated spacecraft is intended to convey the view of the spacecraft as "just" another node on the WAN. In this view, the spacecraft can be accessed as one or more processors, perhaps on an internal LAN, which are accessible using standard communication protocols and network applications. For example, problems and management might be performed by a systems management tool that has access to "current" status of the onboard components. Similarly, data could be retrieved either via client-server applications or simply by FTP. Furthermore, direct migration of applications, e.g., diagnostics, could be readily supported.

This concept requires the availability and implementation of IP or other standard protocols for access. It will generally also need to be treated as a "remote wireless" station, in the sense that continuous communication is unlikely. This implies also the implementation and use of standard architectures and software onboard. Further "on-demand communication" is needed to ensure performance and success. As a result, this concept is likely to be one of the later ones implemented. It does offer one of the most powerful approaches to full integration and automation, which may prove to be a major driver despite the difficulties.

A.3.2 Integrated Ground Station

The integrated ground station (GS) concept involves the ability to configure functionality at the ground stations to optimize WAN performance and acquisition reliability. In general, it should also include integration of the GS as another node in the network, supporting remote management and control, and automation of processes that are manually controlled today. Ideally, the GS becomes simply a reliable pass through for data transport from the spacecraft to the end repository and for commands to the spacecraft. The concept includes such items as self-management for antenna pointing and data configuration, and locally managed fault diagnostics and recovery.

This concept must include options for both dedicated ground stations and those shared as a part of a network. For shared operations, a mechanism for prioritizing to support on-demand communications would be required.

A.3.3 Integrated Customer/Infrastructure Operations

The concept of integrated customer and infrastructure operations is the next follow-on to high-level automation of infrastructure operations. Basically, the intent is to cover all normal operations at a single site that is usually customer operated. This would be supported by the automation of infrastructure operations and provision of agent-driven customer data operations to free the customer from the need for detailed knowledge of the infrastructure. In this mode a generally knowledgeable customer could create plans and drive acquisition of data based on goals and objectives, and the infrastructure would generally respond appropriately. This concept offers what is anticipated to be a high-value product to the customer, near independence from institutional support and operations.

This concept is linked to the ability of COEs to access the infrastructure for diagnostics and maintenance in support of the customer. It would be expected that this is linked to the widely distributed infrastructure operations approach below, at least for maintenance and fault diagnostics.

A.3.4 Widely Distributed Infrastructure Operations

This concept simply applies the concept of the extended network to include all the infrastructure operations. This allows for distribution of major operations functions in multiple locations. This would support separation of normal operations from locations of in-depth expertise, as well as operational splitting of individual functions. There are many possibilities included in this, e.g.,

locating data production at a customer or remote site with total transparency or using remotely distributed monitoring components in COEs. It includes the possibility that the GS can be effectively integrated, without restricting the location of the GS. It also would be a component supporting integration of the spacecraft into the network.

This concept carries with it the implication of high bandwidth capability for the network, with the capacity depending on the specifics of the distribution and the mission data content. It also requires effective network security so that these functions can be accessed over a WAN without breach of necessary security.

A.3.5 Autonomous Infrastructure Operations

Autonomous infrastructure includes the automation (i.e., elimination of the need for human operators) for all infrastructure operations. In practice, there will be an evolving level of automation. It is expected that many of current "nominal" operations can be automated by incorporating one or more intelligent process approaches to resolving issues. Examples of currently developed applications include GenSAA/Genie and OS/Comet for ground scripts to automate processing; Altair MCS for finite state modeling to automate S/C anomaly diagnostics and recovery commanding. Many others exist. Many more can be built in by incorporating failure recovery technology available today with backup maintenance.

This approach requires the acquisition of expertise about the performance of the infrastructure and the ways of handling the various states the infrastructure can obtain. This implies extended development of such knowledge, either during mission implementation and/or during the early mission operations. Standardization of the infrastructure will make more of this applicable to multiple missions, and so more cost effective. This concept is essential to achieving significant improvements in the operations phase costs.

A.3.6 Customer Driven Operations

This concept is closely coupled to the integrated infrastructure and customer data operations concept above. It implies that the entire system is driven off the customer's objectives and plans, and that infrastructure operations are developed to support these plans. The intent is that customer needs are the primary drivers, with exceptions only according to protection of resources. In fact, the customer should be able to choose between limited resource lifetime and otherwise unobtainable information so that the mission returns can be truly optimized.

The concepts discussed in integrated operations provide some of the mechanisms to implement this concept. The intent is to provide a high-value product to the customer that provides control over the use of "his" resources. Implementation of this concept requires that infrastructure operations processes have knowledge of the customer operations priorities and have the ability to incorporate short-term (at least) plans into the infrastructure plans. For example, collecting calibration data would be organized around planned customer operations, subject to maintaining the needed accuracy and/or conditions of the infrastructure.

A.3.7 Common I&T and Operations Phase Systems

The development and use of common integration and test (I&T) and operations phase systems is more an implementation concept than an operations concept. The only real operations impact is that I&T activities could more readily be used to train staff and applications in the behavior of the system under development. This concept can be implemented in two ways.

I&T and operations systems may have *literal* commonality, meaning that the hardware and software used for I&T is then also used for flight operations. However, this approach increases the problem of using "old" equipment during the operations phase. There has traditionally been a long lead time between the specification of GDS equipment and its deployment for operations. The use of I&T equipment for operations increases this time. (This problem is mitigated if the GDS life cycle is shortened.) This approach has additional problems for the cases of multiple spacecraft missions or combined mission operations because then one spacecraft may be in I&T while another is operational.

A less constraining approach is *functional* commonality. In this approach, the same software is used for I&T and operations, but independent or upgraded hardware is used for operations. This approach provides functional commonality between the two systems: the differences are transparent except for performance and the labels on the hardware boxes.

A.3.8 Reliable Communications Protocol

Reliable communications protocols include the provision and operational impacts of a truly reliable protocol for acquiring data from the spacecraft to the ground. It includes on-board storage management as well as transmission to achieve the desired objectives. It would include automated recovery of data lost in transmission by retransmission and preservation of macroscopic data structures during the transmission. The implication is that level-zero processing would be covered by this automated approach. Data accounting would automatically be provided. On-board memory and data storage would automatically be managed to avoid loss of data and overrun of storage space. It is anticipated this reliable transport protocol concept would include transport both of files and real-time data.

This protocol requires the agreement of spacecraft builders and ground system implementers on a common approach. For the protocol to be cost-effective, it must use commercially produced hardware and software. The protocol usage needs at least to cover transport between the ground station and the spacecraft. Strange perversions of standard protocols should be avoided because they eliminate the possibility for commercial support. There are plans for something like this in SuperMOCA. SuperMOCA has the idea of "customizing IP," which would make it impossible to support most of the network applications. Hope springs eternal for someone to recognize that the low level (layers 1&2) could be customized to handle the noise and latency characteristics of space-ground links, leaving the higher levels alone to achieve COTS support.

A.3.9 Spacecraft Autonomy

This concept incorporates several of the previous and following concepts into a global approach that says spacecraft should be much more capable of operating without contact with the ground.

JPL favors this since they are looking at planetary missions where constant communications are problematic. It may also be applicable in low Earth orbit missions, but the relative cost trades are less clear. In general the concept includes automation of the spacecraft performed infrastructure operations, including data management, long-term operations control, and fault isolation and recovery. It presumes an intelligent spacecraft able to make relatively complex decisions or able to fall into a reasonable fail-soft state in the event of problems.

A.3.10 On-Demand Space-Ground Communications

This concept is linked to spacecraft autonomy. It says that the spacecraft should be able to signal the need for communications and establish the space to ground link in coordination with the ground segment. It also allows for the ground segment to similarly overlap spacecraft activity to establish communications when needed. The extent to which this capability exists and to which communications are immediately available affects the cost trades for spacecraft autonomy. The limiting case for this is continuous communications, which requires little on-board automation. The X-ray Timing Explorer (XTE) is an example of using almost limitless communications capability.

This concept requires the ability of the spacecraft to initiate communications. It also means that the communications network cannot operate purely in a scheduled mode, without impacting performance. This has long-term implications for the capabilities of the space-ground communications networks.

A.3.11 Multimission vs. Dedicated Mission Operations Options

There are two basic approaches to operating mission spacecraft: providing a separate system for each mission spacecraft and providing a single system for multiple spacecraft. For a relatively complex mission, it has generally been the choice to implement a dedicated GDS to operate a mission spacecraft, e.g., the great observatory series. The questions are when is it more effective to combine operations. When the spacecraft are highly similar, require close coordination, and/or are not demanding of operations support capability, it can make sense to combine operations into a single system. Particularly, for the New Millennium concept of many microsatellites operating to achieve a single mission objective, many of the above guidelines are met. Ideally, the architecture planned will support either approach.

To combine missions it must be possible to clearly isolate information and operations status for one spacecraft from that of another. Likewise, combining information must be clearly managed. This will increase the development cost of a system by some amount. Similarly separate systems require separate hardware and make operations commonality more difficult. Either approach is achievable today, but cost-effective approaches will need optimizing.

A.3.12 Multispacecraft Missions

This concept is derived from New Millennium, though it also is similar to the original Earth Observing System (EOS) program concept in some ways. The intent is that multiple spacecraft be operated in a coordinated program to accomplish the desired objectives. The intent is to

reduce cost by simplifying the spacecraft and achieving very low-cost spacecraft implementation.

The integration requirements and operations implications depend on how many of the other aspects are incorporated into any such system. Use of process distribution and high levels of automation may allow for integration at a customer site only, with all the rest being arbitrarily distributed. The integration might only be in the form of integrated planning, an operator interface that can view information from all the active spacecraft, and an integrated database of the resulting acquired data. Presumably system management, maintenance and consulting COEs could have visibility into any or all of the elements of such a system, without having a single process manage all data streams simultaneously. Alternatively, a process could be created that handles all streams, sharing all resources among the mission spacecraft data and operations. With care, the operations impacts of these approaches would be similar, freeing implementation from dependence on the operations concept at an early level.

A.4 Cost Modeling

Cost modeling needs to be applied from the early stages of the first phase through implementation. The infrastructure should provide enough information to reliably model mission cost and schedule, in support of trade analysis of implementation and operations alternatives.

Appendix B. Second Generation Mission Requirements

B.1 Purpose

This provides a summary of mission requirement expectations for the second generation Renaissance architecture study. It is based on a combination of specific missions targeted for second generation systems, potential needs for missions yet to be specified, and known trends in spacecraft and ground systems technologies. No allocations of requirements to system components, space or ground, are made in this document.

In general, it is known that the trends are towards more on-board capability and autonomy and towards lights-out ground system operations. Low price missions are also a well-established trend that is expected to continue.

The document provides a preliminary definition of the mission requirements profile to be used to drive the development of a generic architecture framework for second generation systems. As such, both the common requirements and the variability's are captured in preliminary form. The requirements are defined from a customer perspective and have been stated in a form intended to be as implementation independent as possible.

B.2 Operations Phase Requirements Profile

B.2.1 Customer Data Operations

This requirement is for the system to enable the planning, execution and evaluation of customer data operations for the space system. There may be more than one customer and more than one instrument or spacecraft involved in such data operations.

Operations include all activities that deliver data to a customer data product:

- Operations to calibrate elements of the instrument
- Operations to validate correct operation of an instrument feature
- Operations which acquire data of primary scientific interest
- Operations to obtain auxiliary data sets required for successful analysis

B.2.1.1 Plan Customer Data Operations

Plan customer data operations to maximize achievement of objectives. Develop a strategic plan for the life of the mission. Develop tactical plans for each phase of the strategic plan. Assess objective achievements against the cost in mission resources and lifetime.

Plan the specification of a customer data operation. Schedule the data operations. Planned operations can occur anywhere from near real-time to the arbitrary future. The requirement is for development of a plan that includes all elements of the system required to achieve the desired result. The elements required will depend on the implementation of the system and the nature of the data operation.

B.2.1.2 Perform Customer Data Operation

The activity consists of controlling the system to perform the planned customer data operation, including:

- Setting the system state for sensor operations
- Collection of data
- Transport of collected data to the desired location(s) for science operations
- Provide quality assurance techniques to manage data quality
- Perform any subsequent activities required to complete the data operation, e.g., transition to a non-data operation state.

In today's terms, this includes commanding the spacecraft and payload, configuring the ground data system, controlling the transmission of data from the spacecraft to the ground, transport of the data to the center for science operations, and performing data production for the defined product types, all for collection of customer-required data.

Commanding mechanisms include batch submitted uploads, and near real-time uplink connections, with uplink verification data returned. Ground system control mechanisms will include pre-planned, on-line and demand driven control.

Basic data product types include any or all of the following:

Primary data types

- Direct data transfer: real-time IP data transfer of data as received, telemetry channels separated by virtual channel (VC), or data type. Data types include telemetry and tracking.
- Playback transfer file: file for the playback data transferred during a single on-board memory download, with header containing summary information on the data transfer. May contain one or more VCs, depending on the mission design.
- Real-time pass data file: file containing all real-time channel data acquired during a single contact, includes a header containing summary information on the data transfer. Normally a single VC.
- Platform conversion data set: dataset containing the parameters needed for platform parameter interpretation.

- Tracking data file: file containing the complete tracking data set acquired during a single tracking contact, includes a header identifying the tracking system units and information on the specific contact.
- Tracking conversion data set: dataset containing calibration data for tracking system units and conversion algorithms for processing tracking data.

Secondary data sets

- Converted platform data set: platform data converted to engineering/science parameters. Includes orbit, attitude and subsystem information. May have separate files for each type or be merged if all data are in the platform telemetry.
- Added source data sets: data sets from other sources, e.g., star catalogs, planetary ephemerides, earth geologic data sets, et alia.

Tertiary data sets

- Merged data sets: data sets formed by merging and sorting any of the primary and / or secondary data sets into a single set with separately specified boundaries, e.g., for a fixed time period. Includes interpolation of continuous parameters to common time values.

B.2.1.3 Validate Customer Data Operation

Validate the correct performance of the planned operation, including performance of the system components involved in the operation and validation of the resulting customer data. In general, the concept is for the system to perform the operation and notify the operations staff if there are operations problems that are not automatically corrected.

B.2.1.4 Customer Data Operations Response Time

Perform activities to meet customer critical response times, either for portions of the data operation cycle, or from beginning of planning to receipt by the customer.

It is expected that response time requirements will range from roughly 1 minute for critical operations responses, e.g., during early orbit activation sequences, to hours or days for normal data taking operations.

B.2.1.5 Data Volumes

Provide resources to distribute the data volume associated with customer data operations. Volume metrics should include the volume distribution on a per operation basis and an operation frequency distribution.

The maximum anticipated is equivalent to EOS AM-1, which is 150 Mbps for 10 minutes or roughly 5 GBytes in 10 minutes for image data, once per orbit. Lower ranges are typically 5 - 20 Kbps for housekeeping data, similar rates for tracking data, and 100 Kbps to 2 Mbps for playback science data.

B.2.1.6 Customer Data Quality

Provide products which meet the acceptance and goal levels of data quality, where data quality includes metrics on both the collection statistics and on state variable quality constraints.

A typical metric is BER < 10**-9 (TBR) for collected frames and 90 percent of all frames to be collected over any 1 week period. (Probably need a better metric for most missions)

Also included are quality requirements for support data, e.g., accuracy requirements and timing granularity for attitude and orbit or star catalogs and planetary ephemerides.

B.2.1.7 Customer Operations Complexity

Comply with the operations constraints inherent in the customer data operations. For example, a simple operation could be to turn on instrument starting at a given UTC for a specified period, and collect the data as a set. A complex operation may be to configure the instrument in a specific state, point the spacecraft to a specific target, or collect data in real-time for a minimum interval to allow confirmation of the process, followed by an extended observation period with interrupt states when the earth obscures the target.

B.2.2 Manage Customer Information

Provide overall management of customer data types. This includes data production, data storage, data retrieval and distribution. Generally, insure availability of data to customers for their use and provide accounting of the information managed for the customer.

Data production covers production of standard, customer-specified products. Algorithms for production are generally customer specified, and possibly customer developed. Data quality assurance is assumed to be a part of the production algorithms.

Data storage provides for the permanent storage of customer datasets in accessible form. Customer datasets include all standard products and all those developed by the customer and selected for general storage. Storage duration and retrieval requirements will include total volumes, any age dependencies, and availability requirements. Data quality maintenance requirements are another aspect of this. Provide for data security.

Data retrieval covers access to the data by the customer for analysis and product generation. Non-functional requirements are defined in terms of number of users, frequency and type of access, response time requirements, availability to users, and update response requirements. Manage user access to the data to minimize chance of accidental or deliberate destruction of data or denial of service.

Support creation of product types in varying media for distribution. Manage distribution of such products.

B.2.3 System Resource Management

Manage the activity of the components of the system to meet the objectives of the mission. This is described as a single integrated requirement, with the understanding that the design may

partition this into management of individual subsystems at some level. In particular, the system resources may include more than one set of ground equipment, be widely distributed geographically, and include more than one spacecraft.

B.2.3.1 System State Determination

Provide the information defining the actual state of the end-to-end system and all its components at a past point in time. This is intended to include spacecraft state and ground system state. Time can be any past time from near real-time to any historical point.

Components of the system state may include the following. (TBR)

Spacecraft state:

- Platform state: position, velocity, orientation, angular velocity, radiation balance, acceleration vectors, etc.
- Subsystem states: for all subsystems including spacecraft control

Ground data subsystem state:

- RF subsystem state
- Terrestrial communications network state
- Mission management subsystem(s) state
- Customer operations subsystems state

B.2.3.2 System State Validation

Provide evaluation/identification of valid states.

Provide specification of all allowed transitions between this state and any other valid state.

Compare test state with allowed states. If invalid, identify required transition(s) to possible valid states.

B.2.3.3 Manage Resources to Support Customer Data Operations

Plan use of resources to meet customer data operations. Allocate and configure resources for support. Manage operational fail-over or recovery from operational defects to support on-going operations. They include operations planning and commanding of the spacecraft, and planning and configuration management of ground resources. This can be considered as planning and generating state transitions.

This also includes managing resources to meet required data quality. Data quality includes telemetry acquisition, environmental management, and accuracy of subsidiary information, e.g., spacecraft attitude and orbit.

B.2.3.4 Manage Resources Against Loss of Capability

Identify conditions that may lead to loss of operational capability. Provide operational changes to avoid loss of capability, either through permanent damage to resources, or temporary incapacity during operations. Protect resources against accidental or deliberate destruction or denial of service (security).

B.2.3.5 Manage Resources for Future Use

Plan resource needs for future customer data operations. Provide for changes to operational capability through changed resources, changed procedures, or changed staffing. This includes establishing training requirements to achieve desired results.

B.3 Developmental Requirements

Support integration and test of the spacecraft. Provide the required interfaces during this phase, and provide the functions unique to this phase. Support validation of full operation by utilizing the functions described in Section 2. Provide functions to support test operations, capture test data and analyze the data for spacecraft verification and validation tests. Support development and validation of the mission database defining configuration and calibration parameters. Support test at the launch site. Support transition from test phase to postlaunch.

B.3.1 Integration and Test

Integration and test support includes the following:

Provide for an interface to the spacecraft for two-way data exchange. Provide access to the test configuration at the desired S/C integration site. Provide interfaces and support for instrument integration using instrument specific equipment.

Provide for simulation capability during integration to support interfaces with components not yet integrated. Provide for simulation of external interfaces and components, e.g., customer or ground station interfaces.

Provide supplementary tools for detailed analysis and diagnostics during test. Provide tools for temporary reconfiguration during test for diagnostic purposes. Provide for automated test plan implementation and for capture of activities and data from the test. Support specialized interfaces during test, e.g., for thermal-vacuum tests.

Provide support automation to minimize the need for continuous manual intervention in the test process. Support also the ability to repeat precisely the test conditions for any specific test in support of problem assessment and correction. Ensure that the automation does not impede flexibility by human operators when such are active in the test environment.

Support integration and test of the end to end system, including provision of temporary interfaces and test operations activities, and capture of test data and activities. Support validation of decision support subsystems.

B.3.2 Training Support

Support operations staff training. Provide both simulated and actual spacecraft data, and the ability to stage simulated operations for training. This may include shadow-mode operations for training staff.

B.3.3 Launch & Early Orbit Support

Support pre-launch test, launch and early orbit operations, including access to additional staff as appropriate to spacecraft operational configuration and check-out.

B.4 Maintenance

Provide for testing capability, preventive maintenance, repair, replacement of consumables, and long-term enhancement and upgrades.

Provide test support functions for system changes, including both ground and flight components. Support to include provision of simulated and real spacecraft data and the ability to test the integration of the revisions.

Provide planning support functions for assessing the need for modifications to maintain or enhance current operational capabilities.

B.5 Metrics and Variability

B.5.1 Performance

B.5.1.1 Response Times

These requirements apply to any activity. The following are a definition of classes of response times.

- Real-time: Activities must be completed within a fixed time limit, usually subsecond
- Near Real-time: Activities must be completed in near real-time non-fixed times, i.e., subsecond (or seconds for end-to-end times)
- On-line: Activities must be completed in times small compared to human interaction times, i.e., typically less than 5 seconds.
- Off-line: Activities must be completed in times constrained by overall operations only, e.g., sustained throughput.

B.5.1.2 Customer Data Operations Sizes and Frequencies

Specification of the size, frequency and time distribution of required data transactions

B.5.2 Risk

B.5.2.1 Risk Impact Levels

The following are classes of risk impact levels appropriate to NASA-supported missions.

- Manned Space Operations: failure of operations places human lives at risk
- National Resource Related: failure of operations places a National Resource at risk of permanent damage
- Mission Incapacitation: failure of operations places the mission at risk of permanent loss of capability seriously impeding success of the mission
- Secondary Driver: risk is considered a secondary factor to other requirements drivers, e.g., cost

B.5.2.2 Risk Probability

This section defines the probability of occurrence of any identified risk.

B.5.3 Data Quality

B.5.3.1 Allowed Fraction of Customer Operations Data Loss

This includes total fraction, fraction for running time interval, and/or largest allowed contiguous gap.

B.5.3.2 Allowed Undetected Contamination of Customer Operations Data

This includes total contamination fraction, fraction for running time interval, and maximum for any single data operation, or type of operation.

B.5.3.3 Accuracy of System Stated Data used in Producing Customer End Products

This depends on the type of customer data operation to which the state data applies. It includes accuracy of specific parameters, and the accuracy and duration of time interval for which the value applies.

B.5.3.4 Allowed Fraction of Time Excluding Customer Data Operations

This includes both total time during mission, fraction during a running time interval, and/or during any single customer data operation.

B.5.3.5 Quality of Sensor Environment Required for Acceptable Customer Data Operations

This defines the average and worst quality for a single customer data operation, for a running time interval, and/or total of mission.

B.5.4 Constraints

B.5.4.1 Cost

The primary driver for defining mission classes for future missions appears to be cost. The MIDEX, SMEX and University classes are essentially defined by the cost caps and allow any arrangement within that cost envelope.

MIDEX:

- Cost Cap for development: includes development of the instrument, the spacecraft and the ground systems (with some exclusions) through launch plus one month - $70M
- MO&DA cap: all mission and science operations (exclusive of GSFC MO!) - $15M
- Note: Launch costs appear to be outside the caps. Use of GSFC MOC and space-ground communication networks appears outside the caps.

SMEX:

- Development cost cap similar to MIDEX - $35M

University:

- Development cost cap - $5M

B.5.4.2 Operational

Required operations constraints, includes spacecraft autonomy requirements and staffing constraints. Includes use of constrained or shared resources, e.g., commercial communications or existing space-ground communications networks.

B.5.4.3 Geographical

Constraints on locations of ground stations, operational facilities. In general, the profile requirement is to enable support for widely distributed operations, and enable efficient configurations for closely quartered systems. In particular, the system should support migration of functionality to the on-board environment as an operational decision.

B.5.4.4 Mission Unique

Includes mission-specific constraints, including constraints on spacecraft orbits, etc., to allow desired data operations, e.g., must be able to have line of sight with selected geological or celestial targets with a given frequency and duration. The profile requirement is to provide for

mission-specific configuration of tools and controls for effective operation of the specific spacecraft, and to provide mechanisms for efficient addition of components to meet unique requirements.

Appendix C. Architecture Alternatives and Analysis

During the development of the Renaissance architecture several architecture frameworks were defined and evaluated for use. This appendix describes the three primary alternatives defined and the analyses that led to selection of the framework used in the architecture.

C.1 Framework Definition

The frameworks primarily addressed variations in the means of integrating the mission system, providing for data distribution and interprocess control. There were three primary alternatives defined and one that proved to be a subset of the others. The three primary frameworks defined are the hierarchical server, the mission server, and the mission domain.

C.1.1 Hierarchical Server Framework

The hierarchical server framework was designed to address widely distributed mission systems composed of heterogeneous subnetworks, which might not belong to the missions and therefore be outside mission configuration control. Figure C-1 illustrates the primary features of this framework. To accommodate heterogeneous subnetworks, the mission system components are integrated at the global level by a set of data servers, providing data distribution and a common interface between the subnetworks. This framework also includes subnetwork servers for distribution of data to the applications software and direct application control.

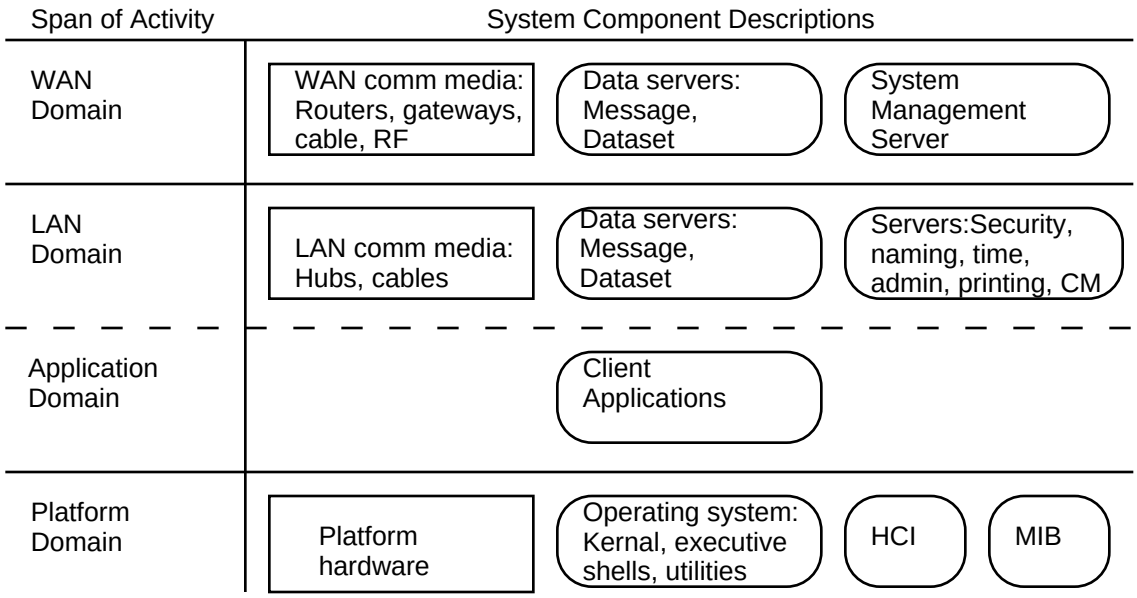


Figure C-1. Hierarchical Server Framework

The processes at any level are presumed to have the identity of other processes at that level within the configuration. Thus, applications would generally only need the address of the local server. The local server has access to all applications and to the contact location of the global server. The global servers in turn have the addresses of the other servers in their domain. Two general types of data servers are included, message servers to provide time-critical distribution and dataset servers for bulk data interchange.

System management is designed as a single consolidated process for the mission system. It is recognized that there will be elements outside the control of the system management for which information is needed. These are treated as externals to the system management process, providing an interface that delivers status information to the system management process, while not generally accepting control directives.

C.1.2 Mission Server Framework

The mission server framework is focused on simplifying the design as appropriate to mission dedicated systems. In this case, the WAN level is restricted to one form or another of LAN interconnection. Essentially, the entire system appears as a single logical LAN. Figure C-2 shows the layering model for this framework.

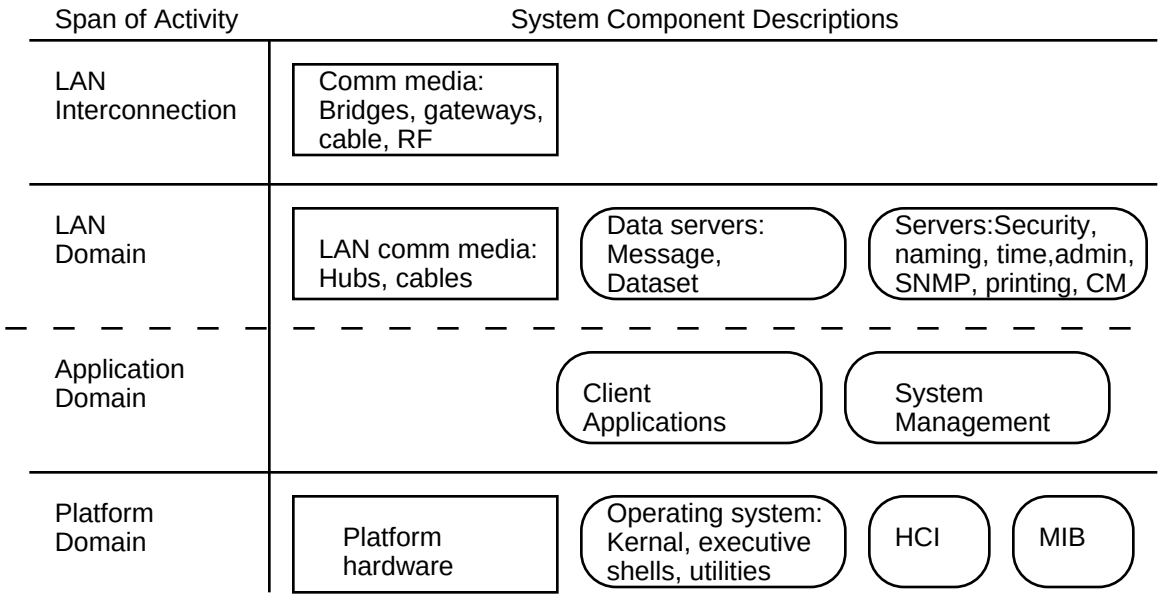


Figure C-2. Mission Server Framework

In this alternative, the center of initiative is based in the operations center. All data servers are in the LAN domain, and specifically, all in an operations center. The concept is that the operations center servers are provided specific addresses to connect to ground station applications for data exchange. This architecture most closely resembles that of today, in that there is no general WAN domain information base. Message level and dataset level servers are separately provided and have client applications in the operations center.

As seen in the domain view, there are no processes with WAN-level domains. System management is primarily a center-based activity, with the exception that the GS is run as an adjunct to the operations center. The system is generally not knowledgeable of other operations centers, given the lack of WAN level information and must be explicitly connected to exchange information.

System management is again a single process, but there is not the same problem with external segments because of the mission dedication concept. All elements are under the control of the mission system management process.

C.1.3 Mission Domain Framework

The mission domain framework is a hierarchical model in the domain view, with only one WAN level process, the Domain Directory Server. Otherwise, full peer-to-peer relationships are supported. Not all data exchange needs to be mediated by a data transfer server. Figure C-3 illustrates the domain hierarchy for this model.

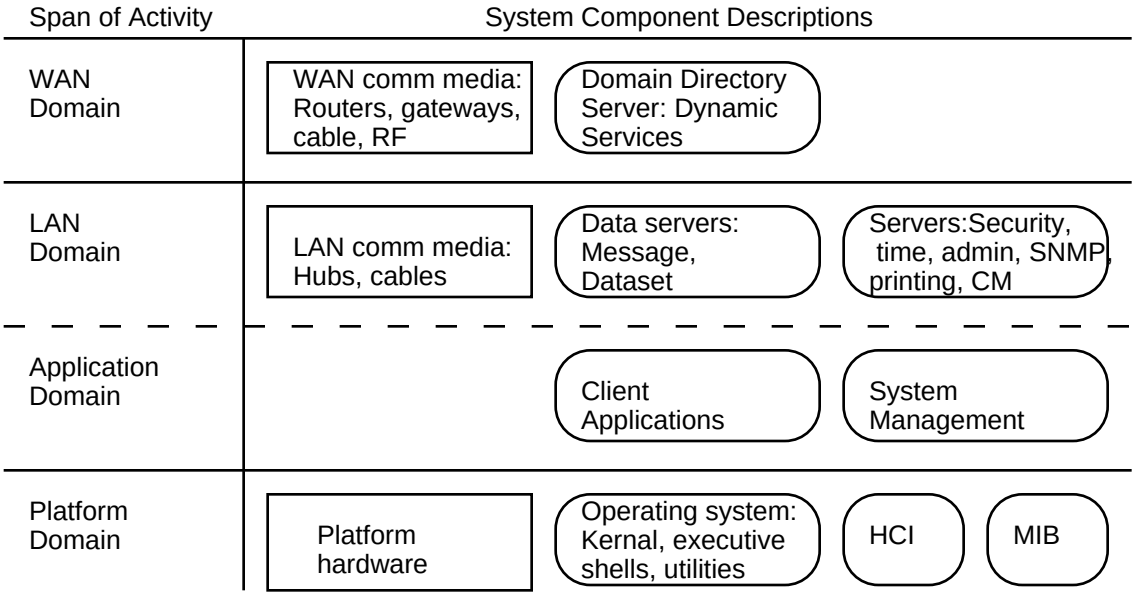


Figure C-3 Domain Services Framework

The two major differences between the mission server framework and the mission domain framework are the extension to a true WAN through the use of a domain directory that points dynamically to globally available services and the addition of a separate API for server to server communications. The separate API allows for use of different LAN implementations, providing that the common APIs are maintained.

The Domain Directory Server is a generally accessible process with dynamically managed information about all resources on the system. With this information, any authorized process can acquire the information to connect to other processes as needed to perform an activity. The dynamic nature of the name service allows for recovery if an active process or other component

fails. System management is an adjunct to the naming service, providing status information and supporting recovery.

In this alternative, most of the data transfer functions at the ground station are seen as servers, supporting multiple clients at any time. Clients may be true client applications or other servers. As seen in the domain view, applications can be intended to operate with local LAN-level servers or with remote servers. Any application in any operations center can establish communication with these servers, supporting direct telemetry receipt or commanding as needed. Similarly, any application can connect to information sources in an operations center from any WAN-level location, making support operations an easily integrated element.

System management is treated here as a true distributed process, providing integrated management of components over the entire network even though the management application(s) reside(s) within a LAN domain. This is accomplished through publication of management event messages to the Domain Directory Server to ensure the currency of the information on the server.

There is a management process for each operational entity with management responsibility. Actual resource status is available through the Domain Directory Server. The management application controls configuration for those elements within the region of responsibility. For systems with extensive WAN segments, it is expected that the WAN will have its own management process for that segment.

C.2 Analysis of Alternative Frameworks

The analysis of the defined frameworks is separated into two parts: an assessment of the virtues and the drawbacks of each of the frameworks and a synthesis from these of the most desirable approach to be followed. In this synthesis, elements of each were selected for desirability and compatibility, and combined into the desired framework. Thus, the resulting framework is not identical to any of the three.

C.2.1 Assessment of Virtues and Drawbacks

C.2.1.1 Hierarchical Server Framework

This framework provides a good mechanism for integration of widely varying elements. The separation of the WAN servers from LAN domain services allows for varying implementations at the LAN level within the architecture. These variations can include security constraints, performance needs, and unique environmental needs, e.g., for a complete real-time environment. Additionally, the hierarchy of servers avoids the need for system-wide knowledge of all available processes, which can be a burden in dynamic systems.

A disadvantage is the inverse of its virtues: the architecture is quite complex for a very simple system or for the spacecraft I&T version with no need for extended distribution. There are performance issues that could raise costs unnecessarily for simple systems.

The server concept allows varying implementations with differing cost, performance and availability where this is controlled.

Centralized system management provides a very effective mechanism for ensuring the system responds to mission activities properly. Conversely, it may be very difficult to implement in an environment where much of the extended system is composed of shared resources under differing management. It requires treating these as externals. A related problem is that centralized management of a large system can be very difficult because of scale. Both acquiring the needed status information and performing the needed analysis can be very resource consuming. For example, standard SNMPv1 can cause a significant data load on a WAN if managing a far flung network of components.

C.2.1.2 Mission Server Framework

This framework provides a simpler design approach that effectively meets the needs for early phases of mission systems, e.g., spacecraft I&T and for small dedicated mission systems. It eliminates one entire layer of the hierarchical server framework. This resembles the approach taken today. It provides complete, consistent mission control over all elements of the system. This will greatly improve the ability to manage system performance over today's approach.

The lack of a WAN integration structure poses challenges for integrating elements over long distances, e.g., a remotely located ground station. It is not clear that the approaches that are suitable for LAN operations will work effectively over long distances. As long as data rates are not too large and SNMP traffic is not too great, this should be fine.

There is a general issue of integrating multiple environments, e.g., shared resource segments such as DSN, which do not allow for centralized control. This framework provides no simple mechanism for conflict management, since the operations centers are not really integrated. Users can adapt to this by treating shared resources as an external entity, which is essentially how things are done today.

With this model, integration with applications in a support center, e.g., a COE concept, would be more difficult, requiring that each system be given the explicit addresses for those organizations supporting it. Control of access would be in the hands of the operations center, not the support location. Again, the COE will be treated as an external entity.

Finally, there is no apparent mechanism to support heterogeneous environments within the system. This will limit the ability to truly integrate the spacecraft, or any other true real-time environment at the LAN level. Such integration would have to be addressed at a lower level, limiting the level of complexity that can be addressed.

C.2.1.3 Mission Domain Framework

This framework provides a different mechanism for integrating a distributed system, though still server based. The WAN integration and LAN integration are provided by separate APIs, with a global directory service to allow any process to identify and connect to appropriate resources across the system. The required server at the WAN level is the directory server. It acts as an information broker, supporting peer-to-peer and client-server connections across the WAN. This allows the WAN system to be treated more simply as a single integrated network yet provides more flexibility for managing differences between LAN cells.

The single largest disadvantage of this approach is that there are no current implementations of this. There are a few X.500 servers just becoming available that might provide the Directory Server implementation. There are few, or no, applications and system management tools that can use such a mechanism. This concept would fit a CORBA-like implementation, with CORBA 2.0 interfaces across the WAN, but there is no CORBA implementation for the Directory Server at the WAN level. Because implementation requires both the Directory Server and the processes to use it, there is a high uncertainty as to when, or if, such implementation will occur.

It is not clear what rules are required for mediating communication between peers. Does the Directory Server keep the status of connections between processes? Are all applications required to have a function supporting multiple simultaneous connections? In general, at the WAN level, peer-to-peer is complicated because of the potential need for any process to manage an arbitrary set of contacts. All such processes are required to manage both the many types of contact and the numbers of them. It is not clear that many data server implementations exist to support peer-to-peer connections, though certainly some exist.

C.2.2 Framework Synthesis

C.2.2.1 Data Distribution and Interprocess Communication

As part of the analysis performed by the architecture action team, both server-based and peer-to-peer pipe and filter approaches were analyzed for streams of data messages such as telemetry. Based on this analysis, it was concluded that the server-based approach was more generally applicable, while the pipe and filter peer-to-peer approach allowed better performance for the same applied resources. In analyzing the frameworks, the server-based approach is judged better for WAN and LAN level data distribution and IPC.

The rationale is that the server can manage several forms and streams of data distribution providing a higher level of configuration flexibility that is desirable for those domain levels. Peer-to-peer pipes would seem appropriate at lower levels, e.g., for a real-time subdomain. As the number of components increases, the potential interactions increase as the factorial, making the peer interaction software much more complex. By keeping the interactions pair-wise, the client-server approach reduces this, and further keeps the contact management software within the server.

For data set transferals, both server-based and peer-to-peer relationships can be used, and peer-to-peer is more commonly implemented. Common processes already exist for file transfers of varying types, and there is little reason to exclude any of these. For databases, transfers are generally done in a client-server approach, and current implementations meet our primary needs.

The result is that servers are the approach taken for most data transfer at the WAN and LAN levels. IPC is similarly approached for the same reasons. There are potential performance issues, but current evidence is that these can generally be managed at a subdomain level and with appropriate allocation of hardware resources.

Two general classes of data servers are identified. The message server supports data stream types of data distribution as well as other message types. It is intended to provide time-critical

exchange. The dataset server class can be time-critical but more frequently is not. Dataset servers may exchange all, or parts, of a pre-existing dataset. The implementation will depend on the situation.

C.2.2.2. WAN/LAN Services

The flexibility of the hierarchical server approach is given a high weight due to the expected need to build mission systems using shared resources and incorporating LAN environments with differing implementations. In particular, the desire to incorporate the spacecraft as just another LAN cell on the network seems to drive a need for varying implementations. Near-future missions will frequently want to use common resources and be distributed across the continental US, at least. This approach allows a common architecture integrating everything from the spacecraft through the end customer's payload analysis operations.

There is a need for simple and inexpensive implementations for localized systems. These include spacecraft I&T and mission systems such as the proposed SMEX-lite series. For these missions, the WAN level is not applicable. In the case of I&T, this may be a temporary factor, in that system I&T may require that the spacecraft I&T become part of a larger network. For these reasons, the interface between LAN level and WAN level is allowed to be an option. All critical services must be able to be applied at the LAN level. This allows the incorporation of the best features of both the hierarchical server framework and the mission server framework.

One clear implication is that system management must be implementable at the LAN level and must allow for the existence of more than one LAN cell. For most missions, consolidated, central system management is not possible. Use of shared resources precludes this. For widely distributed cells, it is not generally desirable. In particular, if the spacecraft is to be a cell, it is desirable for that cell to have localized system management capability when it is out of communication contact at a minimum. The latest, and near future, releases of commercial system management tools will support distributed management so that implementation is not a major risk. The result is a design approach that supports federated management at the cell level, with the ability of a given cell to provide configuration directives to remote cells for general system configuration needs.

In general, the interprocess communication and the system management processes are perceived as event-driven processes. Schedules can create time-driven events, but not all events are schedule driven. In particular, all elements should be allowed to incorporate event driven systems, such as UNIX operating systems, to allow for concurrent processing activities and use of standard operating environments. This will significantly change spacecraft control when this change is implemented. It is noted that the SCL product currently under evaluation for spacecraft use is generally event driven, and supports concurrent processes.

Global directory services, as identified in the mission domain framework, are highly desirable but are not widely available. The increase in flexibility provided by this is very effective in supporting abstract service support as implemented in DCE. The lack of general market support would raise the implementation cost and risk significantly. However, this does represent a potential evolution path for the architecture.

C.3 Message and Data Servers

A key feature of the design is its use of message and dataset servers at the WAN and LAN domain layers. These servers act as intermediary processes that provide reliable delivery of messages and datasets from producers (sources) to consumers (destinations). Use of these servers eliminates the need for multiple point-to-point connections between producers and consumers. Instead of requiring a separate connection between every producer and consumer, each producer and consumer need only connect to the server. The server receives messages and data from multiple producers and sends or makes available the messages and data to multiple consumers. This approach reduces the number of interfaces that must be implemented between processes,and makes it easier to add new applications to a configuration.

C.3.1 Message Server Types

A *message* is a packet of data containing information that must be communicated promptly. Data delivered as messages include telemetry frames and packets and operations event data. Message servers handle the delivery of distinct messages from producers to consumers. Message servers generally provide mechanisms for selective routing of their contents to consumers. These include messages targeted to a specific consumer, selection by message class, and selection by message content. Distribution can be to single consumers or groups.

A *dataset* is a collection of information that is generated for use at an arbitrarily later time. Typically, datasets are larger than messages, but this is not always the case. Data delivered as datasets include software images, data logs, and customer products. Dataset servers handle the distribution of complete or partial datasets from producers to consumers. Dataset servers provide either interactive access to the dataset (such as a database management system or a file sharing server like NFS) or physical transfer of the data to another location (such as a file transfer server like ftp).

C.3.2 Server Interaction Styles

There are a number of standard server interaction styles in use. The three styles desired for near term implementation are described in this section.

The primary differentiator among the three major types of servers is the means of determining the destinations of messages and when they are sent. Some current COTS servers implement only one of the capabilities described below, while others implement several.

C.3.2.1 Distribution Servers

With this type of server, message producers determine which consumers will receive each message, and when they will receive it. There are several types of distribution servers, as described in the following.

• **List Distribution Server** – This type of server distributes each message to a list of recipients specified by the producer of the message. Destinations may also be specified as distribution lists that are managed by the server.

An example is a **mail transfer agent** (MTA), such as sendmail. A Message Distribution Server is used in this Renaissance design to distribute schedules, status reports, and other information messages to members of the operations teams in the operations centers.

• **Multicast Server** – This type of server is implemented with a group of server processes called a *process group*. A message producer addresses a message to the process group. The members of the process group cooperate to ensure that the message is reliably distributed to all members of the group. Several process groups may be defined. The message producer selects the multicast destinations of the message by sending the message to the appropriate group. The process group may also ensure that messages are received by each member in causal or other orderings.

An example is the Isis product. (With Isis, servers can dynamically join and leave process groups. Therefore, the destinations of multicast messages are also partially determined by the recipients in a publish/subscribe fashion.) A Message Multicast Server is used in this Renaissance design to multicast messages reliably across the WAN.

C.3.2.2 Request/Response Servers

With this type of server, message consumers determine what messages or data they want. They receive the messages or data in response to individual, specific requests.

Request/response servers receive and store or process messages from producers. Clients make specific requests of the server and receive responses that contain or are based on the stored or processed messages. These types of servers can be used in both the WAN and LAN domains. There are several types of request/response servers, as described below.

• **Message/data managers** – These types of servers store and manage the messages or data that are provided to requesting clients. Several types of message/data managers exist: the newsgroup server, the playback server, and the database server. A newsgroup server receives and stores distinct messages received from message producers. Clients request the messages by group name. A playback server receives and stores distinct messages in time sequence. Clients request that messages be sent to them in order, with optional timing characteristics. A database receives and processes messages, and stores the processed data in tables, objects, or named variables. Clients request the data by retrieval keywords. A database is used in this Renaissance design to provide access to orbit, attitude, schedule, and other application data.

• **Brokers** – These types of servers pass requests received from clients on to other processes that retrieve or produce the requested data. The other process sends the requested data back to the broker, which passes the response back to the original client. Current broker types are the Object Request Broker (ORB) and communications facilitator agents.

An ORB receives requests from clients and routes them to server objects in conformance with the CORBA standard interfaces. The ORB finds the server object in the network; the client need not know its location. ORBs may be used in future Renaissance designs to provide a location-independent interface between COTS products.

Communications facilitator agents help requesting processes find other processes that can provide some needed information or service. They provide communications facilitation functions that are more advanced than ORBs. They may be used to support advanced automation functions in future Renaissance designs. Examples include:

– **Broker/Advertise agent:** the agent is engaged by a client as a broker to look for processes that advertise a needed service or information. When a server is found, the agent sends it the request, receives the response, and passes it to the client.

– **Recruitment agent:** the agent is engaged by a client to recruit a server that can perform a needed service or provide needed information. The agent helps establish the initial relationship between the client and the server.

– **Recommendation agent:** the agent is engaged by a client to find and recommend a server. The client and server establish the relationship.

C.3.2.3 Publish/Subscribe Servers

With this type of server, message consumers determine what messages or data they want. Consumers receive requested messages or data in continuing streams, receiving new messages or data updates as the server receives them from producers or periodically.

The server receives subscription requests from clients for various classes of messages. When the server receives a message from a producer, the server distributes it to all clients having an active subscription. Publish/subscribe servers are primarily used in the LAN domain. They typically handle a large number of message categories. There are several types of publish/subscribe servers, as described below.

• **Event Message Server –** This type of server distributes streams of distinct messages to clients who have active subscriptions. An event message server is used in this Renaissance design to distribute operations event messages and directives to processes in the operations center.

An example is the Email list server. A list server receives and optionally stores distinct messages received from message producers. It sends a copy of each message received to all members of a distribution list. Clients add and delete themselves from the list. Multiple distribution lists can be maintained. An example is the listserve program.

• **Named Variable Server –** This type of server is often called a Data Server. It distributes streams of named variable updates to clients who have active subscriptions for those variables. A data server manages a current value table and can provide the current value of a variable on a periodic basis (asynchronous service) or whenever its value is updated by the producer (synchronous service). A data server can also have a request/response capability to allow clients to make a one-shot request for data items without requesting a full subscription. Some data servers maintain a history of values of variables. Clients can retrieve the history to perform near real-time trend analyses.

A named variable server is used in this Renaissance design to distribute telemetry values and state variable values to processes in the operations center. A current product providing this is the Talarian RTserver**.**

C.3.2.4 Summary of Servers in Design

Table C-1 lists the types of servers that are included in this design. For each server listed, the table identifies the domain (WAN or LAN) it serves, indicates whether it is a message or dataset server, and gives examples of possible implementations. Depending on the COTS products selected, several of the server functions listed may be supported by the same product.

Table C-1. Summary of Servers in Design

Server	Domain	Type	Examples
List Distribution Server	WAN	message	sendmail
Multicast Server	WAN	message	Isis
Dataset Distr. Server	WAN	dataset	ftp
Dataset Distr. Server	LAN	dataset	DFS, NFS, AFS
Data Base	LAN	message, dataset	Oracle, Sybase
Event Message Server	LAN	message	RTserver
Named Variable Server	LAN	message	RTserver
Object Request Broker (future)	LAN	message	HP, DEC ORB
Communication Facilitator Agents (future)	LAN	message	KQML agents

C.4 Graphical Interface Alternatives

Generally, the user interface for an application can be constructed as a part of the application or as an independent element. Constructing the user interfaces as separate processes is currently viewed as having some important benefits: support for process distribution, easier maintenance, better support for common interface features. Several models for these independent user interface processes have been used historically, ranging from a single, monolithic interface process for all applications to completely separate processes for each application. The Renaissance second generation architectures being considered are composed primarily of COTS software components. In such configurations, data presented to the user may be generated by several different products. These configurations may present an inconsistent graphic user interface (GUI) that is fragmented with obvious seams. Further, different applications may take different approaches to separation or integration of the user interface. Several approaches can be taken to attempt to mitigate this problem.

As increased levels of operations automation are incrementally built into Renaissance GDSs, the user interface will be used less for routine monitoring and control activities. A user interface will still be required, however, to support nonroutine activities that are not yet automated, such as the resolution of new anomaly types or replanning for unanticipated configurations. In this

environment, user interface consistency and ease of use will be even more important. This is because operations staff will interact with the system less frequently and will not have the opportunity to learn peculiarities of the user interface through extensive use, as is the case today.

Most of the discussion below assumes a UNIX environment. It should be noted that the X-Window System provides a mechanism for distribution of the display processes independent of any application design.

C.4.1 Independent Motif GUIs

In this alternative **(#1)**, every application generates its own display windows. Applications are required to use the X Window System (currently X11R5) and conform to the OSF/Motif standard. OSF/Motif has already been widely adopted by vendors for applications that run in the UNIX environment.

Pros:

a. Simplicity – This is the simplest alternative for implementation. All COTS we have investigated claim to be X and Motif compliant.

b. Basic similarity in look and feel – If an application is Motif compliant, it implies that its GUI:

Uses the standard Motif widget set

Has been developed using the Motif Style Guide

Displays its windows using the Motif Window Manager.

These three items define the basic look and feel of the GUI.

Cons:

a. Format inconsistency – The format of displayed data may be considerably different in windows from different applications, but many applications support user customization allowing changes to achieve a more standard look and feel.

b. Seams – The display windows are clearly divided between applications.

C.4.2 CDE Conformance

In this alternative **(#2)**, every application generates its own display windows, but all applications are required to conform to the Common Desktop Environment (CDE) standard. CDE is an attempt by UNIX workstation and software vendors to create a standard user interface environment for UNIX users, such as those that exist for Macintosh users and MS Windows users. CDE was announced in 1993 by the Common Open Software Environment (COSE) consortium with wide vendor support. It builds on the most prevalent user interface technology available for UNIX systems at that time.

The CDE standard involves several elements, including:

X11R5

OSF/Motif widget set and window manager (Release 1.2.3)

X Window *Inter-Client Communications Conventions Manual* (ICCCM) - defines how X applications should communicate with one another

CDE User Interface Style Guide - an expanded version of the OSF/Motif Style Guide

Workspace manager (HP VUE) - provides a graphical interface to the UNIX desktop

DeskSet (SunSoft)

ToolTalk (SunSoft)

Windowing KornShell

Two levels of CDE conformance are possible: *basic integration* and *full integration*. These are treated separately below. In general, it is expected that a COTS product will be built to meet one level or the other without modification.

C.4.2.1 CDE Conformance: Basic Integration

CDE *basic integration* compliance means that an application is installed on the CDE desktop and that it can be launched from icons that represent the application or data files associated with the application. Vendors can typically make their COTS product compliant to CDE at this level without making any code modifications to the application. It requires that icons be constructed to represent the application on the desktop, and several files be built to define icon behavior and help information. Most UNIX COTS applications are expected to have CDE basic integration compliance in the near future.

Pros:

Launch and help uniformity – Applications can be launched in a consistent way and have help displayed using a common CDE help manager.

Cons:

a. Format inconsistencies and seams – The cons of Alternative #1 still apply.

C.4.2.2 CDE Conformance: Full Integration

CDE *full integration* compliance means that an application implements all the required features of CDE, including drag and drop between applications, print services, internationalization, and runtime help. Significant coding changes may be required to make an existing application CDE compliant at this level because of the CDE specific functionality required. Although vendor support for CDE is strong, there will likely be delays before some COTS applications achieve full integration compliance.

Pros:

a. Additional behavior uniformity – Drag and drop works across applications; full on-line help is provided for all applications in a consistent way, print services for all applications.

b. Standard fonts - Full CDE compliant applications will generally use fonts in the standard CDE font set, which will be available on all CDE platforms. This will avoid the font problems that often occur in multiplatform environments.

Cons:

a. Format inconsistencies and seams – The cons of Alternative #1 still apply.

b. Vendor support for CDE is expected to be widespread, but some products may lag.

C.4.3 Display Manager

In this alternative **(#3)**, the data generated by all applications is displayed by a common display manager. The common display manager ensures that all data are displayed consistently, and seamlessly. Because applications do not generate displays, they are separated from the GUI. Two major approaches are possible: a data server, and a display server. They are treated separately below.

C.4.3.1 Data Server Approach

In this approach, all applications generate data and send it to one or more data servers (or data bases). A common display manager receives the data from the data server (or retrieves it from the database), formats it into display windows, and presents the windows to the user. The format and content of the windows are described in display definition files that are written in a display specification language such as user interface language (UIL). Users request display windows by display name and may not be aware of what applications generate the data displayed.

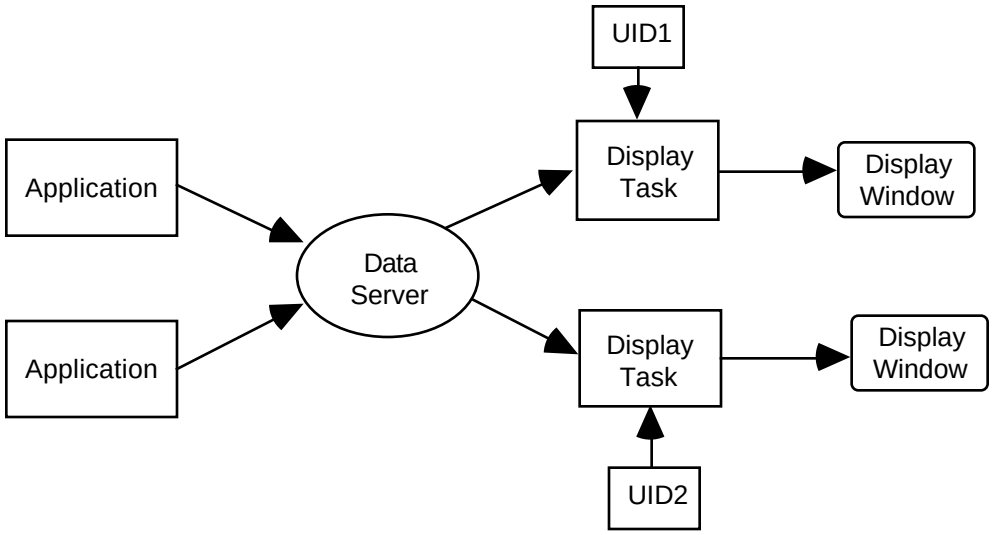


Figure C-4. Data Manager: Data Server Approach

With this approach, data from different applications can be seamlessly combined on a single display. New applications can be added to the configuration by having them send their data to the data server. COTS products can be added if they have an API that conforms to the data server or can be wrapped to conform to the data server.

A version of this approach is used in TPOCC at GSFC, in the Integral Systems EPOCH 2000 system, and in the Multimission Advanced Ground Intelligent Control (MAGIC) system being developed by the Air Force Phillips Laboratory. In the TPOCC design, a variety of applications send analog and discrete-state variable values to a data server; applications include telemetry decommutation, command, simulation, history, and others. Each display window is created and updated by a separate instance of a display task. Data to be displayed are sent from the data server and an event server to the display tasks. Displays are defined in UID files, which are compiled from UIL descriptions. User input that manipulates the display windows is handled directly by the display task. User input that controls applications (directives) is sent to the TSTOL process.

Pros:

a. Seamless consistency in format – All display windows can be defined to format data in a consistent manner. Displays can be defined that combine data from multiple applications.

b. Real-time updates – Can easily handle real-time display updates if the display manager receives the data from a data server in a publish/subscribe manner.

c. Multiple platforms – Because the applications are separated from the GUI by the data manager, displays can be adapted to different windowing environments and platforms without modifying the applications. For example, the display manager could display windows on NT workstations while the applications execute on UNIX servers.

Cons:

a. Display definition – Definitions must be created for all displays, using e.g. UIL.

b. Display manager development – Parts of the display manager component may not be available as COTS.

c. COTS API requirements – All COTS applications must provide APIs that enable their output data to be sent to the data server. Wrappers may have to be built to adapt the COTS API to the data server interface.

d. Limits on applications – Some useful COTS applications may not be included in the system because of the cost or impracticality of integrating their output with the data servers in this way.

C.4.3.2 Display Server Approach

In this approach, a user employs a display client to view and interact with displays that are provided by a display server. In a typical interaction, the user requests a display from the display server. The display server determines which application (or data base or data server) is responsible for producing the data required for the requested display, and communicates with the application via gateway processes. The gateway uses the application's API to send it the request and receive its reply. The gateway then generates a display based on the reply data, and sends the display to the display server. The server passes the display to the requesting user. All requested displays are generated in a consistent format. Multiple gateways can be written to interface to different applications. A gateway could retrieve data from more than one application, and generate a combined display.

A popular example of this approach is the use of a World-Wide Web (WWW) server with Common Gateway Interface (CGI) programs. WWW clients ("browsers") are available for all common platforms, and present displays in a consistent manner. Popular browsers include Mosaic and NetScape. Displays are described using the Hypertext Markup Language (HTML). Users send display requests and parameters to the WWW server using the WWW "forms" interface. CGI programs (gateways) are written in C, or in any of several scripting languages (perl, Tcl, shell scripts, etc.). Each gateway can be tailored to interface with the API provided with a desired application.

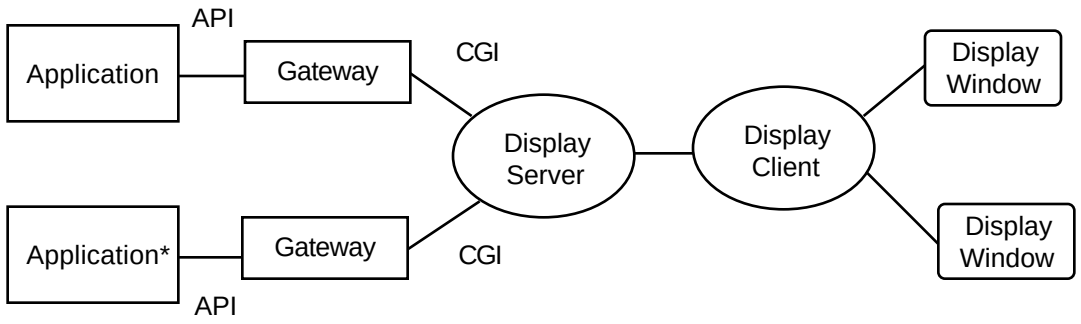


Figure C-5. Data Manager: Display Server Approach

This approach was used in the HST operational trend analysis display system that was completed in May 1995. The system receives a request from a Mosaic user, retrieves HST trend data from a database, generates a plot using the PV-WAVE COTS package, and sends it back to the Mosaic user. This general approach was also used for the COSTLESS project prototype that demonstrated how WWW technology can be used to provide a common user interface to legacy software systems. The prototype allows a Mosaic user to view and modify data managed by TPOCC.

Pros:

a. Seamless consistency in format – All display windows can be defined to format data in a consistent manner.

b. COTS flexibility – A variety of COTS packages can be used, each potentially having a different API. COTS are not constrained to producing data that can be stored in a data server.

c. Multiple platform support – Because the applications are separated from the GUI by the data server, displays can be viewed on any platform and windowing environment for which there is a display client.

d. Image support – If the application generates a pixmap image, e.g., a plot, the gateway can incorporate the image into the final display with other data.

Cons:

a. Gateway program development – Gateway programs or scripts must be written to generate the displays using data provided by the application via its API.

b. Real-time updates – Best suited for situations where each display is requested by the user. Not as well-suited for situations where displays are dynamically updated with real-time data.

> (Note: Emerging extensions to HTML, such as Sun's Java, will better support dynamic display content. Java is a programming language for describing the dynamic behavior of display pages. Pages are described with HTML and small Java programs, called applets. The page and applet are sent to a browser that contains a Java interpreter.)

C.4.4 Hybrid Approach

The hybrid approach combines alternative #3 with alternative #1 or #2. In this alternative, a display manager is used to handle a core set of operations displays that provide a seamless GUI. However, some additional applications are also used that generate their own Motif-compliant or CDE-compliant GUIs. Both types of display windows can be displayed simultaneously on a display screen.

Pros:

a. Advanced display applications – Applications that generate unique displays can be included in the system by using their display windows directly. Such applications include advanced data visualization tools for 3D physical model rendering and abstract data representations.

C.5 System Configuration Alternatives

This section describes three alternative approaches to designing processes for managing the configuration of systems for space data operations. The focus is on ensuring that the correct software, hardware and datasets are properly configured to operate spacecraft with a high level of automation and reliability. The intent is to meet the full need for managing many kinds of

simultaneous activities, with the option of multiple spacecraft, in an integrated system. The three approaches defined are script-driven configuration, state-driven configuration, and event-driven configuration. Each is described in a separate subsection below. Each section contains the summary description of the primary features and differentiating elements, and a summary of the initial assessment of the benefits and drawbacks of the approach.

C.5.1 Script-driven Configuration Control

This approach uses a defined script database to explicitly control configuration of processes for any given activity. Execution of the scripts causes the establishment of a system configuration. The simplest form of script usage is that establishing a static configuration. A more complex approach allows the evaluation of conditional values during execution to select branches of execution. This form is needed for any ground script usage for automating pass operations. The concept is historically in use and can be implemented in many ways. Use of standard shell scripts in UNIX provides a mechanism with no need for any additional processes to interpret or compile the script.

There are at least two subalternatives within this one: translated scripts (e.g., shell scripts) and compiled scripts. Dynamic translation at execution time is slower than a compiled version but provides for more rapid change. If the system is to be used for the I&T environment, a translator capability is important because of the need for editing changes during tests for diagnostics and recovery. During the operations phase, the number of errors that are truly unforeseen is greatly reduced, and the need for translation may not exist. During the operations phase, consistency is much more important, driving a need for the ability to capture proven scripts and control changes to them to minimize risk to the operations and the vehicle.

The benefits of this approach include

- The simplicity of implementation, at least the basics
- Direct visibility of the directives at all times, and ease of linkage to a scheduled activity
- Flexibility of configuration. A properly designed script capability can perform any activity the operator could perform manually

The drawbacks of the approach include

- There is no explicit configuration management.
- Without a server process, the conditions and options for initiating a script are very limited.
- Scripts are constrained in the logical structures for concurrent configuration control.
- Evaluation of status is constrained to the points and conditions specified in the script, limiting its ability to manage unforeseen conditions.

C.5.2 State-Model Driven Configuration Control

In this approach, the configuration is controlled by a state manager that has information about allowed states and state transitions. The version considered under this alternative is use of a state model and a schedule to drive the issuing of directives for system configuration. The approach is to use the state model and the schedule to drive not only execution, but selection of the directives to be executed. Knowledge of current state and of the target state will specify all the directives required to change the state. The intent is to use a mechanism with more general intelligence to drive the configuration process.

The benefits of this approach include

- More automation of configuration, to include initiation of scripts based on required state transitions.
- More general characterization of required directives based on state model.
- Easier control of the transition, based on the state model rather than reading scripts.

The drawbacks include

- Still only schedule driven, limited information about system is available.
- Evaluation of conditions only undertaken at specified times, limiting responsiveness to changing conditions.

C.5.3 Event-Driven Configuration Control

This approach drives the system configuration by a server that responds to general types of system events, more generalized than the state machine-based version. This explicitly would incorporate full information about current system status, as well as planned states and activities. The approach is to use the server to issue directives, based on analysis of current and planned states with the ability to consider nonoptimal conditions or real-time changes. Optimization of capability can be built in with the knowledge of the system and the operation objectives. Use of a rule-based inference engine and/or a state model to support decisionmaking would also be appropriate. Finally, events that drive the issuing of directives includes the full set of possible system events, including time-driven, data-driven, sequence-driven, error-driven, or externally driven events.

The benefits of this approach are

- Maximizes responsiveness to real system activities and events without manual intervention
- Supports response to unplanned, spacecraft initiated, data transfers
- Provides a standard mechanism to manage both ground and space system changes and problems

- Can be extend conditional execution with the use of an inference engine as a part of the executing process
- Can embed scripts where sequence of directives is important
- Supports ability to automatically manage systems that are event driven, with concurrent processing

The drawbacks of this are

- Not currently a standard approach, except for ground computer networks
- Requires construction of the intelligence during I&T and tuning during early operations
- More complex software to implement the server

Appendix D. Capabilities and Design Drivers

D.1 Functional capabilities

This section describes the capabilities to be provided by the architecture. These are grouped as generic and as phase-unique by mission phase. These capabilities descriptions are an adaptation of the specifications contained in Appendix B, the preliminary requirements profile.

D.1.1 Generic Capabilities

Capture and maintain the spacecraft design information

Capture and maintain the GDS design information

Capture and maintain the data specifications

Develop and maintain automation processes, including scripts, control data definitions, etc.

Support planning of all operations, including operations definition and scheduling, and resource management

Monitor the status of spacecraft and GDS components in a timely manner to keep risk of failure within mission acceptable limits

Provide fault isolation and recovery management for the spacecraft and GDS components

Command the spacecraft and GDS components to enable operations as needed

Prepare and maintain the data and software needed for desired operation of the spacecraft and GDS

Provide communications between all elements of the mission, spacecraft, GDS, and end customer, to meet operational needs

Provide data product generation and delivery

D.1.2 Spacecraft I&T unique

Provide communications and power interfaces to spacecraft components

Perform detailed tests to validate operation of spacecraft components

D.1.3 L&EO unique

Provide communications interfaces to spacecraft launch facilities

D.1.4 Operations Unique

Provide communications and power interfaces to spacecraft components

Perform end to end customer data operations, delivering data products under the direction of the end customer

(Refer to Appendix B for operations details.)

D.2 I&T to Operations Phase Transition Issues

This section discusses the primary differences that must be managed in making the transition from I&T to the Operations phase.

D.2.1 Flexibility

I&T requires a high level of flexibility in configuring the elements of the system used to test components. Changes must be made quickly to adapt to errors detected to test more efficiently and effectively. There is a question about whether there is flexibility because the spacecraft builder will NOT build to meet specifications. If the latter, there can be more serious cost questions.

This requirement is similar to the multimission need for operations phase systems, in that the cost of customizing to a new mission should be minimal, which generally means easily changed and a high level of change possible.

Note that these requirements generally drive the design to support data driven configuration for as much as possible of the system. It can also drive the desirability of effective tools to support configuration data changes and validations.

D.2.2 Configuration Control

The operations phase wants control over configuration to avoid accidental or poorly conceived changes to the system. The rationale for this is that changes made in haste may result in serious damage to the mission, either directly to the spacecraft, or by putting one into an operational state which is likely to result in operational loss, e.g., running out of power, uncontrolled spin, etc.

I&T phase will not see many of these because putting the spacecraft in a wrong state is easily recovered during test phase. If one told the I&T team that human errors were going to cost $500K each, there would be a lot more interest in avoidance thereof.

The net result is that the conflict between flexibility and CM should result in selective control levels, both by data type and phase. Further, the levels should be definable as an operational decision not requiring code changes. This links to system security.

D.2.3 Operations Automation

The general principle is that automation must be cost-effective. It should not cost more than the possible payoff. But automation has implications in more than one area: the obvious one of reducing the number of people required, the potential to reduce the risk of human error, and the potential to perform fault isolation and recovery so as to reduce risk to the mission. Artificial intelligence systems have shown generally the ability to perform some types of analysis more quickly and more accurately than humans.

During the I&T phase, several factors mitigate against automation: the need to develop an understanding of the system before you can automate it, the need for higher levels of flexibility that are costly to implement, and the need for implementation timeliness, i.e., having it ready when it's useful. Note that I&T systems already are fairly automated, e.g., use of STOL procedures to automate test and configuration. The only argument is the level of useful automation.

The stated estimate of cost to develop an information model that was given is based on a separate effort, essentially replicating the effort that the FOT must do to operate the spacecraft. The development or acquisition of tools that would allow for capture of this information as developed during design, I&T, and operations training would substantially reduce this cost. This does not obviate the need to cost justify any automation.

D.2.4 Operational Reliability

This issue relates directly to mission risks and CM above. The problem is that systems that are not reliable in function and performance can put the entire mission at risk. There is also a cost issue, in that any operation that must be repeated because of error is more costly.

For a mission in which contact time is a scarce resource, reliability for both command and telemetry transfer is very important. Experience with some I&T systems has indicated a level of reliability that has risks in an operational phase. In operations, there may not be time to recover from command errors, either during emergencies, or simply because the operation is complex enough to need the available time. I&T activities are much less sensitive to this type of error, because contact time is not so limited.

D.2.5 Availability

Ground system availability is another key operations phase requirement. Most I&T systems are not designed for effective fail-over. They have single points of failure. This may be insufficient for operations phase activities. Specifically, neither the 740, nor the 730 I&T systems appear to have any mode for redundancy of critical elements. Separate, hot backups may be the only choice for redundancy, which raises both operations and cost issues.

D.2.6 Interfaces

I&T and operational systems generally have very different needs for interfaces. I&T has direct interfaces to test equipment and through that to a spacecraft connection. Operations phase has connections to all the ground stations and networks and to the end customers.

For I&T the interfaces tend to be local and to consist of differing standard bus structures, e.g., 422 and 1773. Data formats coming in telemetry are generally consistent, and single, though not all parameters will necessarily be present during early phases. Generally, there is not a need to simultaneously support multiple telemetry interfaces.

For the operations phase, the low-level interfaces are generally standard, e.g., the proposal for a common external interface via IP to the GSFC network. However, there are likely to be multiple interfaces at the high level, which means that different data and different formats are needed. Examples include the varying formats for scheduling and for ground station data from SN, Wallops, GN, etc. Another common variability is the interface to the end customers, who may be using pre-existing hardware and software. The result is that the operations phase system must be able to use the right data formatting and typing for each contact with either a space-ground communication system and customer. The operational system generally doesn't need the instrumentation interfaces, e.g., 1773, but that may change if ground stations become local.

D.3 Infrastructure and Interface Issues

A paradigm is needed that can support the functions described in the previous section that will operate in the environments described in the context and operations options section.

D.4 Security and Availability Issues

The applicable security and availability issues that affect how a mission is operated and the impact of these areas on the architecture must be identified.

D.4.1 Security

The architecture needs to address resource protection (protecting the assets of the ground and space segments from both unintentional and intentional disruptions).

The architecture needs to address data security to ensure that data are not corrupted intentionally or unintentionally, as well as protecting proprietary data (plans and science data) from unauthorized disclosure.

D.4.2 Availability

The architecture needs to address availability in a distributed processing environment. Reliability and maintainability issues need to be considered in developing an architecture that will meet the availability criteria. Other criteria to be used for evaluating the availability of a system should be identified.

Appendix E. University Mission Design

E.1 University Mission Class

This label is applied to a wide range of low-cost missions designed for secondary launch capability, limited spacecraft mobility and payload size, and short duration, aimed at supporting both true science objectives and innovative technology for instrument and spacecraft. The class allows higher risk, based on low cost and frequent flights. For this design, the Spartan-LITE concepts of Ron Pollidan (Code 681) have been used as a baseline for the mission system.

The overall cost is limited to $10M full life-cycle cost, with an operational lifetime not required to exceed 6 months. This avoids the need for formally "space qualified" hardware and support environments, leading to lower cost, but useful, spacecraft. Launch is presumed to occur as Shuttle canister launch or secondary payload for ELV launch, with total launch cost < $1M. Innovative technologies, e.g., GPS or equivalent is assumed. A spacecraft on-board processor of Intel type is assumed. spacecraft construction is assumed to be outside GSFC.

Spacecraft operations are to be performed by the PI, at a PI-selected site. The ground system may be provided by GSFC, but the PI can choose any alternative that meets mission needs.

E.2 Operations Concept

There is one key assumption about operations made for the development of this architecture: that an operations mode can be developed allowing the postlaunch operations to be performed with no "real-time" command and control from the ground.

This is not an unrealistic assumption. For most missions today, any problems will most probably occur while the spacecraft is out of communications. This means that any analysis and response must be out of real-time. The usefulness of real-time becomes limited to providing more efficient operations by providing data transfer feedback, e.g., receipt of valid command uploads. A not too great improvement in data transfer and on-board capability would eliminate the need for this, allowing the system to dispense with real-time requirements.

For the Spartan Lite model of at least 72-hour autonomy, the assumption of no real-time operations seems reasonable. Provision of interactive command and control, with messages exchanged via Email, would allow for the needs of the mission. These Email messages would be routinely stored on the ground with routine analysis triggered by receipt of the message. For manual analysis, an interface such as the WWW might allow a remote or local operator to see status pages set up by the spacecraft for analysis of problems. This could be supplemented by download of selected state history via the downlink file transfers for more complete analysis.

The on-board requirements added include a provision for a somewhat higher level baseline of recovery mode. The baseline would have to support these higher level interfaces. This could require provision of boot ROMs for the computer that include the needed support. These may

need to have added parity bits to support bit error recovery in the memory, but this does not seem very stringent. If SBCs are being used, then a backup computer and ROM could readily be provided to ensure the baseline level of functionality in safe mode.

A related assumption is that I&T elements to support real-time diagnostics of components can still be included for efficiency during I&T but that these components need not be transferred to the operations phase elements and might truly be unique to the spacecraft builder. The only requirements would be that the I&T system include components to build a standard spacecraft specification "database" that could be transferred to the operational environment. Given the likelihood that such mission spacecraft may not all be built by the same vendor, it is not certain that any standard I&T implementation can be expected.

E.3 System Design

On the basis of the operations assumptions identified in Section E.2 and on a preliminary assessment of the objectives of the mission class, an initial identification of mission operations components at the systems level has been made. Figure E-1 shows the system level view of the mission system during operations. It consists of a spacecraft, a ground station, an operations center, and one or more customer sites that may be located close-by or remotely to the operations center.

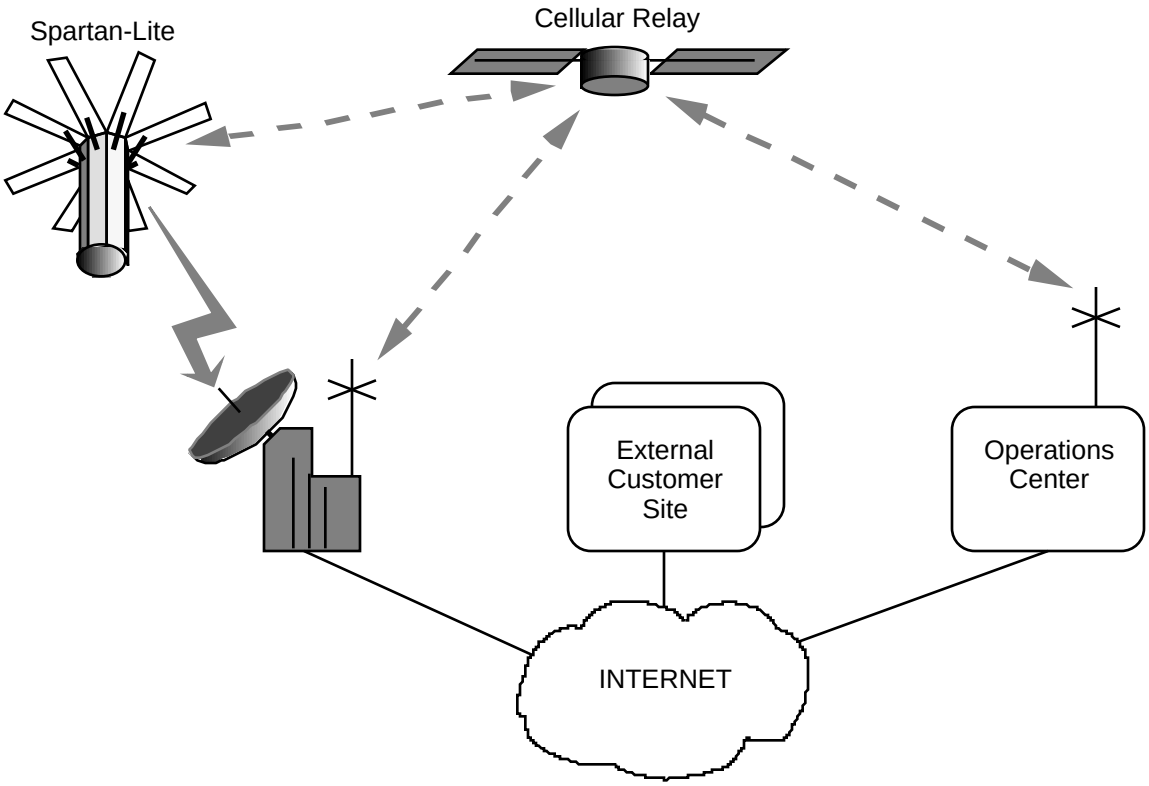


Figure E-1. Mission System Overview

As shown in the figure, the spacecraft connects to the operations center using a 9.6 Kbps cellular phone link for command and control, and downloads data to the ground station directly at 2 Mbps to unload payload data and stored history data. The cellular phone link from the ground station supports return path data for IP data transfer to the ground.

E.3.1 System Level Design

For the operations phase, Figures E-2, E-3, and E-4 show the next level component designs for the spacecraft, the operations center and the ground station, respectively. As can be seen in these figures, the ground station and operations center are built around a common framework, based on non-UNIX PC platforms and a LAN. Network transparency is implemented with a combination of standard IP applications and protocols, and DCE. The spacecraft design is shown to identify applications and servers necessary to form the interface with the ground. Essentially, the spacecraft appears as a single platform connected to the network.

Figure E-2 shows the component interconnections for the spacecraft. The major data related hardware components are shown as rectangles on the top, and the software components as rounded rectangles on the bottom. The two types are interfaced through a PCI bus (or other standard bus). The Programmable Device Controllers (PDCs) are the data acquisition and controllers for the spacecraft and instrument subsystems. These are managed through the Spacecraft Control process, and monitored on-board by a State Monitor process. This forms the essential on-board capability for autonomous operations.

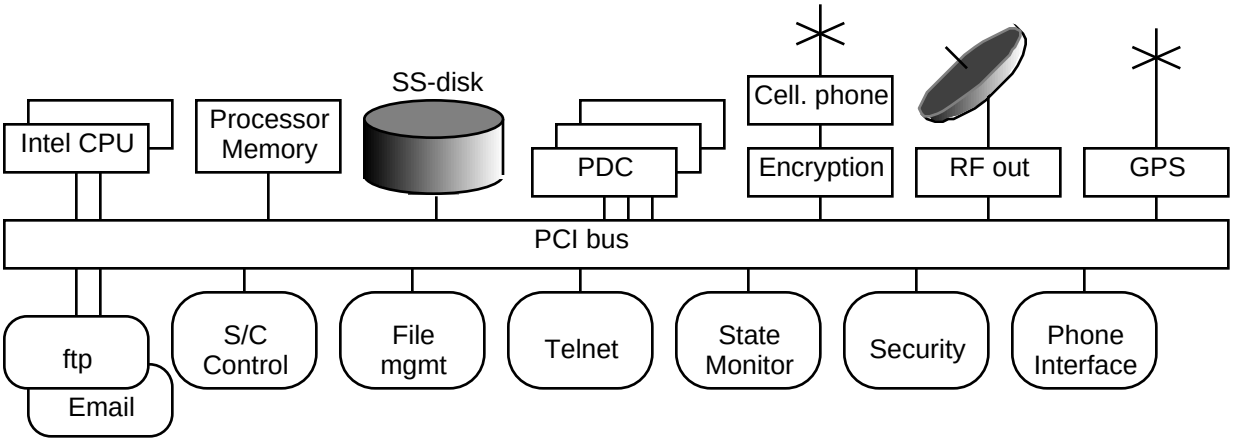


Figure E-2. Spacecraft Interconnection Diagram

The essential interfaces to the rest of the system for the operations phase are the Email, ftp and Telnet applications, adding to the network transparency. Email and ftp provide the standard communications and data transfer mechanisms. Telnet is there only to support on-line diagnostics for anomaly handling. Note that on-board file management is presumed to support creation and management of the on-board solid state memory, designed as a solid stated disk to support use of COTS based data access. Software-based security and hardware based encryption

are provided to manage the difficulties resulting from publicly accessible interfaces over the cellular telephone interface. On-board GPS, or equivalent, is provided to avoid the need for ground-based orbit determination and tracking. This is part of the Spartan-LITE baseline.

Figure E-3 illustrates the interconnections of the software components of the operations center. As shown, the message and dataset servers for this design are primarily standards based: Internet for access to data over the WAN level and DCE based for LAN level servers. The exceptions are the use of a DBMS for local dataset access and the use of DCE WAN services for timing and naming services. The cell has a security gateway to isolate most activity from the public interfaces for data (the WWW server). The telephone interface uses encryption to exchange data with the satellite as an authentication process to prevent commanding by unauthorized sources.

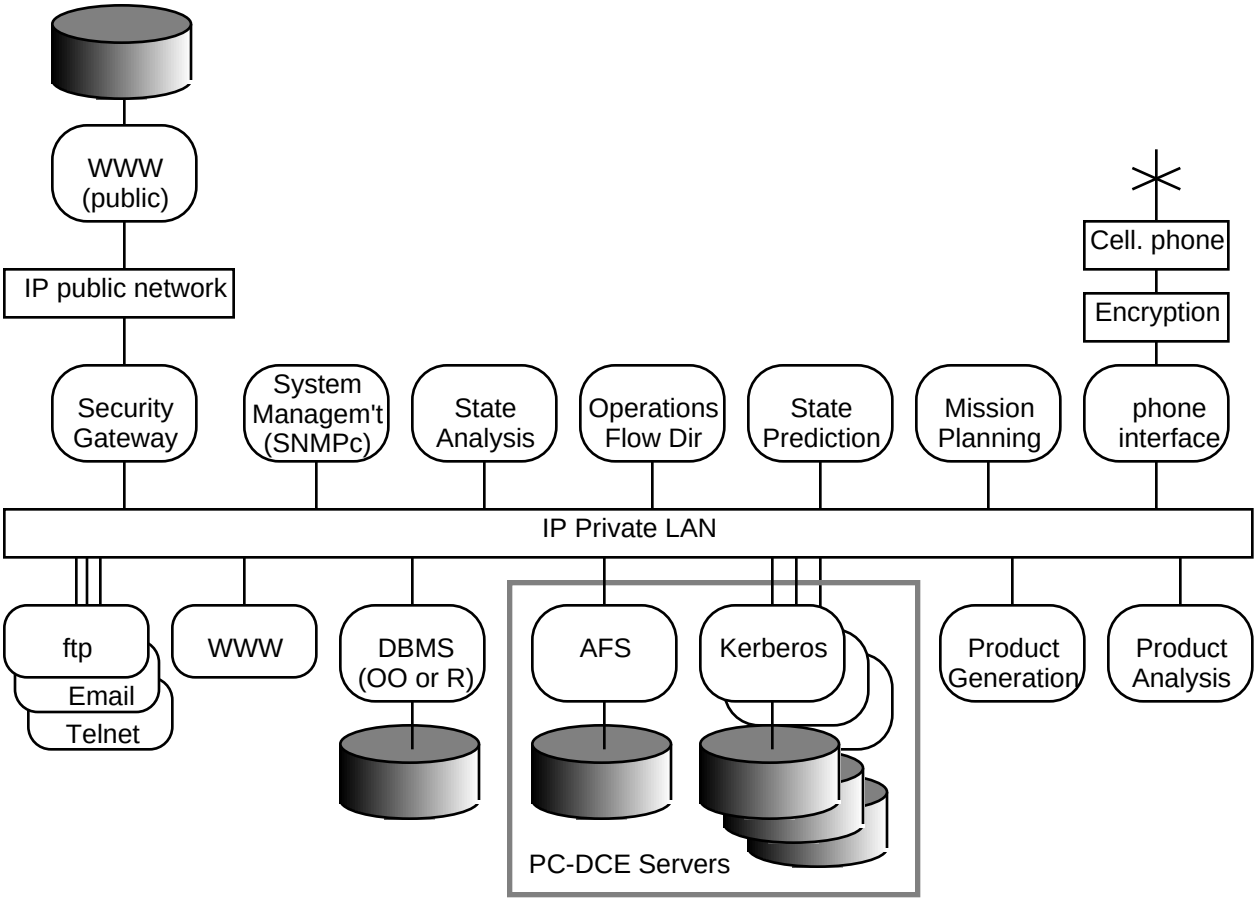


Figure E-3. Operations Center Interconnection Diagram

The monitor and control applications are SNMP based and include state analysis and product analysis for assessment of proper operations, flow control to manage the automated flow of activities, and a system management application for overall system status data management and command distribution. State prediction provides planning support and the mission planning process provides for development of planned activity definitions and scripting. Operations flow

control provides the primary mechanism for scripting responses to data received from the spacecraft and for initiating activities specified by Mission planning. The system management process provides data to the protected WWW server for human access to system information. For more detail, the Telnet application gives direct login capability, using secured access through the security gateway.

Figure E-4 shows the system interconnection diagram for the ground station cell. Hardware and software components are indicated in the same way as for the operations center. The server design is essentially the same as for the operations center, with Internet and DCE servers.

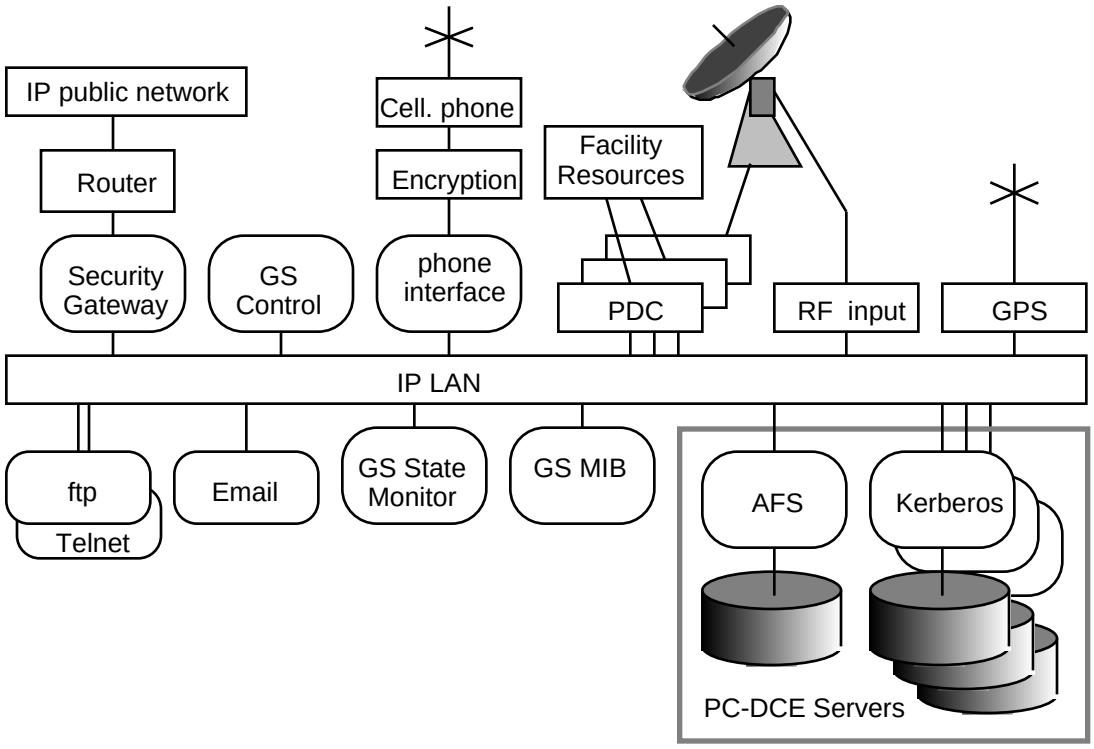


Figure E-4. Ground Station Interconnect Diagram

GS control, state monitor and MIB form the local agent of the system management task, eliminating the need for full-time communications with the operations center. If the ground station is collocated with the operations center, or very close to it, then these may be simplified to allow direct control from the operations center process. Email provides a method for the ground station to receive information from the spacecraft supporting contact management. GPS provides a synchronized time capability linking spacecraft and ground station. The cellular phone connection is also used to support the return path for ftp transfers of stored data from the spacecraft to local storage or the ground station can simply act as an IP gateway for ftp from the operations center if communications is appropriate. The illustrated design is for a remote ground station.

E.3.2 Communication Architecture

Figure E-5 shows the overall communications architecture. As illustrated, the protocols are shown over the large arrows, while the data transported is shown next to the smaller directional arrows.

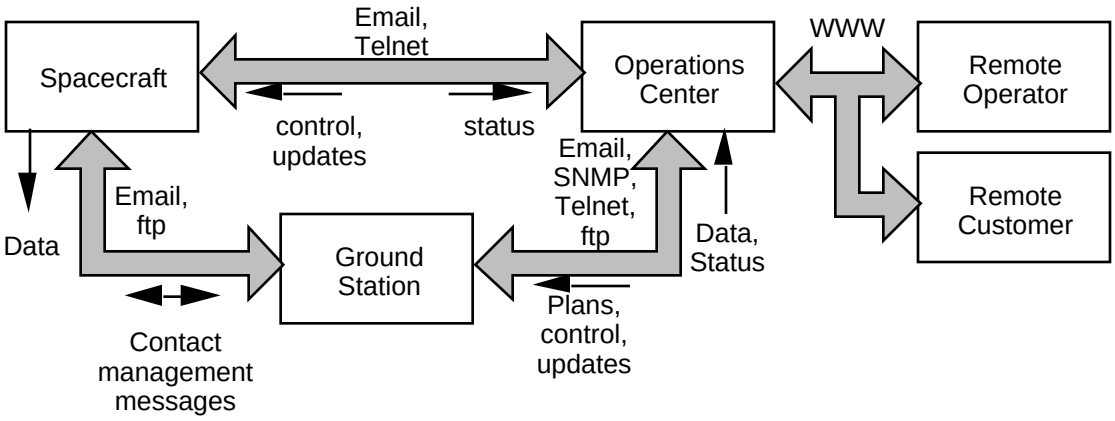


Figure E-5. Communications Architecture

The spacecraft receives control data and image updates via Email from the Operations Center and transmits status data via the same mechanism. Telnet is used as an emergency access to the spacecraft from the Operations Center.

The spacecraft and ground station exchange contact management messages via Email and use ftp to transport data from the spacecraft to the ground, with a return IP path via the telephone link.

The operations center sends control data via SNMP and operations plans via Email to the ground station and receives status data back via SNMP. Image updates are sent via ftp to the ground station. Spacecraft payload and vehicle data sets are also transferred back from the ground station via ftp. Telnet exists as an emergency access to the ground station for maintenance.

The operations center exchanges status and control information with remote operators via the WWW and provides remote access to payload data products through this protocol and indirectly via ftp. Both public and secured exchanges are supported. The operations center also includes the ability to beep operators via the cellular network when human assistance is required.

E.3.3 Software Integration Architecture

The software server architecture is shown in Table E-1, the Domain Hierarchy. The WAN servers are shown in the top layer, the LAN servers in the next layer, the application layer is next, and the platform layer is shown at the bottom. As described before, the wide area servers are Internet based, while the local area servers are primarily DCE based, with some applications having interfaces to both domains.

Table E-1. Domain Hierarchy

Domain	Components	
Wide Area	Email -	SMTP message server. Primary message server for passing control information between the spacecraft and the ground.
	SNMP -	Message protocol for system management data.
	WWW -	HTML dataset server. Primary exchange for status information documents, and product related documents.
	FTP -	Dataset server. Primary server for bulk data transport.
	Telnet -	Message server for remote login to platforms. Used for diagnostic procedures only.
	DTS -	DCE Distributed Time Services. Platform Clock synchronization.
	X.500	Extension to DCE name services (CDS).
Local Area	AFS -	Dataset server for flat files. Used within cell for non-replicated data sharing
	Kerberos -	Security Server. Process authentication and data access control.
	CDS -	Name Services. Provides location independent process name access for applications.
	DBMS -	Database server. Provides query based access to datasets, either object based, or relational.
Application	N/A	
Platform	PFS -	Platform file management services
	Web -	Browser providing HCI for external access.

Within the server hierarchy, several features of DCE are extensions of the DCE cell servers to the system level, notably the Distributed Time Service and the X.500 extensions to the directory service. There are no application subdomain servers in this design. PFS is simply a call-out of the platform operating system file management system, which is universally used as the lowest level access. Lastly, the DBMS is used to store and retrieve structured datasets. Either a relational or an object-oriented DBMS serves this role. Such datasets can include configuration data, status data or even products, depending on the mission.

E.3.4 Applications

The applications specified for spacecraft, ground station, and operations center are designed to function as integrated units to serve the activities identified in the generic architecture. Examples of this are the command and control segments and the production segments.

Figure E-6 shows the set of applications that provide integrated command and control and their interconnections. The monitor and control applications at the ground station and spacecraft act as agents of the system management application. Within the operations center the operations

flow director acts as a scripting agent for planned activities and preset actions, e.g., initiating analysis through the system management application upon receipt of a status Email message from the spacecraft. In the diagram, the disk icon represents the general repository, recognizing the Email messages will be addressed to specific process IDs. Note that the Email messages between spacecraft and ground station are only to coordinate high rate contact operations.

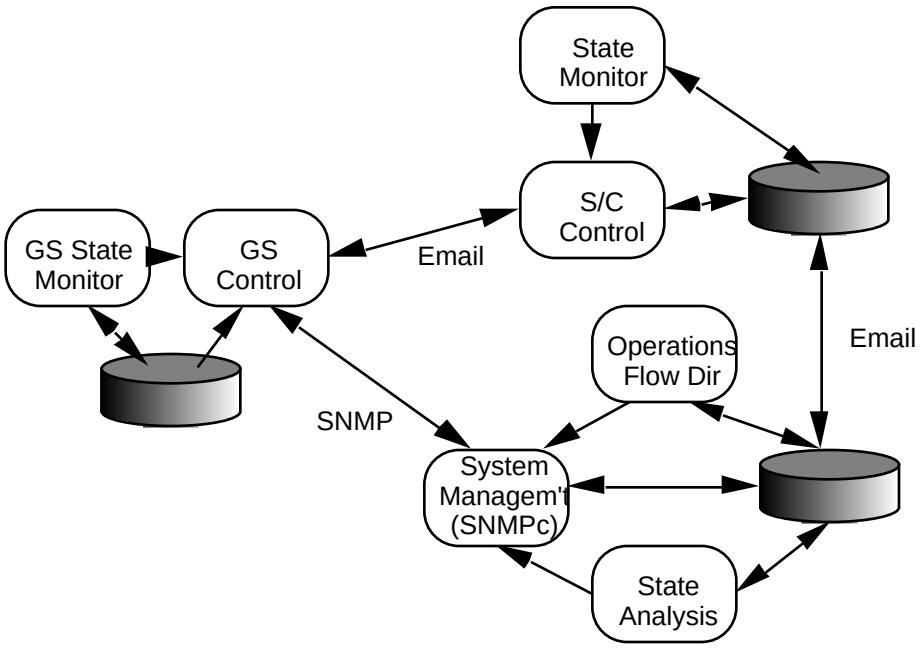


Figure E-6. Monitor & Control Applications

As can be seen in Figure E-7, most of the production flow is performed in this approach by the general-purpose servers. Most functions performed today by "Level-Zero Processing" are not needed because the data are managed as consistent sets and transferred using reliable transport. Product generation produces the specific integrated products desired by the customer, e.g., Level-1 and above, with boundaries set by the customer. Alternatively, the customer could choose to take the sets as received, since they are self-contained with orbit and attitude included.

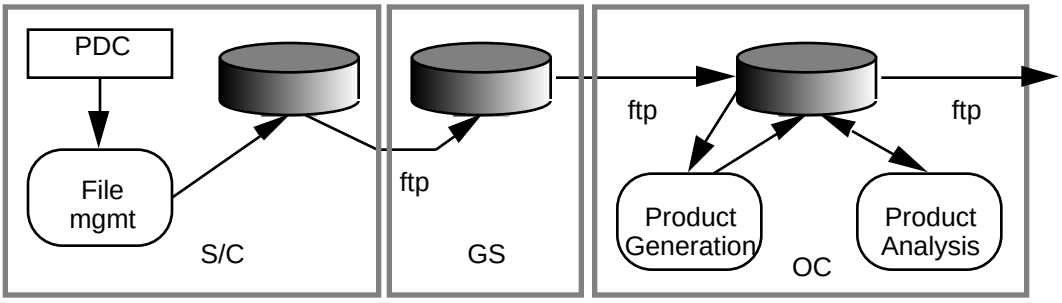


Figure E-7. Production Applications

Table E-2 provides an allocation of the design applications to specific existing products that can perform these functions. All such selections are very preliminary, based only on paper analysis and not yet on prototyping and benchmarking. The design application names are in the column on the left and possible products, with comments, on the right.

Table E-2. COTS Application Allocations

Component	COTS Application
Spacecraft Control	ICS SCL(?)
Spacecraft State Monitor	ICS SCL(?)
System Management	Castle Rock SNMPc
State Analysis	
Operations Flow Director	ICS SCL(?)
State Prediction	Orbit: KKI Orion; Trending: MS Access + MATLAB or PV-Wave; STATLab Pro (ODBC access); Systat
Mission Planning	ICS SCL,
Product Generation	TBD
Product Analysis	TBD
GS Control	MMS application (?)
GS State Monitor	
Security Gateway	High-end options: Harris CyberGuard Firewall, Digital Firewall for UNIX PC-based: Checkpoint Firewall-1 (Solaris)

E.4 Analysis of Design

This sample design stretches the mission implementation in major ways from traditional NASA systems. The intent has been to identify possible paths to much lower cost. As can be seen, this depends on significant changes in how the spacecraft is implemented and operated. The following sections summarize our assessment of this preliminary design.

E.4.1 Benefits

The single largest benefit is that the system is moved entirely into a PC environment with the associated lower costs. By removing the need for real-time functions other than in communications, many existing COTS products can be applied to this problem. Implementing and operating this appear to be significantly lower cost.

E.4.2 Risks and Mitigation

Two major risk areas are present: spacecraft implementation using standard components, and the apparent gaps in our current knowledge of applicable COTS products. The spacecraft implementation issue goes to the heart of the concept for the mission, i.e., the willingness to accept higher risks because of low cost and short expected life times. Selection of applicable COTS products can be addressed by a benchmarking analysis of possible applications, providing clear understanding of the capabilities and possible shortcomings of these products.

E.4.3 Operability

The current operations concept allows for short-term failures of ground system components and spacecraft operations. Within this context, the availability can be enhanced through the use of selected hardware redundancy where off-the-shelf software supports automated switchover or even some manual switchover.

For example, on the spacecraft, one issue is that the spacecraft must have a reliable recovery of communications capability to support ground assistance in recovery from some problems. This can be provided through the use of memory based error recovery, i.e., single bit error algorithms and stored parity bits for this. It can also be enhanced through the use of duplicate processors on-board to avoid fatal processor failure.

On the ground, use of multi-CPU machines and RAID disk arrays for capture of data will enhance reliability of the system, minimizing unrecoverable errors. Similar use for the system management platform will make it unlikely that failures will be unnoticed.

Data quality is enhanced by the use of CCSDS and Reed-Solomon as a data link protocol, and the use of ftp and file management capabilities for product datasets. This combination means that data captured is of high quality, with minimal random errors. Further, most communication dropouts will be handled by providing files at some modest size so that transfer is typically about 1 minute. This way communication drop out boundaries will not cause significant retransmissions.

Security can be handled by COTS products for operator authentication, digital signatures and encryption. The largest threat is probably denial of service because the cellular telephone is busy. For this reason, this design includes spacecraft initiated calls as a fallback to increase the probability of a successful call.

Appendix F. ASIST Oriented Design

F.1 ASIST Design Alternative

This design is intended to support current spacecraft architectures but extend the life cycle of the ground system to include the development of the spacecraft components. The basic ground system architecture is based on the Advanced Spacecraft Integration and Systems Test system (ASIST) developed by NASA/GSFC Code 733. The purpose of the architecture is to provide a control center architecture to communicate with the satellite through different media using the same ground system components at each stage. The resulting architecture should support autonomous and NASA-integrated missions equally well.

F.2 Operations Concept

There is one key operations concept assumption that is addressed by this design approach:

The mission system will grow from a group of disjointed pieces, each supporting some function of the mission, into a whole system. The same systems used in the laboratory to test a spacecraft component will be integrated into the ground system at the same time that the component is integrated into the spacecraft.

As the system evolves, the command, control, and analysis functions for each subsystem will be distributed to a number of workstations, referred to as segments. Each segment will be responsible for analyzing telemetry that relates to its particular subsystem and generating real-time commands for upload to that subsystem.

To ensure continued spacecraft safety, a central controller will monitor the state of the spacecraft. Commands that alter the state of the spacecraft will be evaluated before execution to ensure that the command can be carried out safely. The level of automation and the location of this function will change as the mission system matures, with more human intervention within the ground system during early development and integration phases and more automation (possibly on the spacecraft) as the spacecraft moves into operations. Moving the command mandate onto the spacecraft allows the spacecraft to operate in a more autonomous mode. The impact of commands can be analyzed during the actual operations of the spacecraft (using live data points to determine the spacecraft state) instead of having to be compared against an expected state modelled on the ground.

F.3 System Design

F.3.1 System Level Design

The system level for this design is driven by the assumption that the mission system will be built as a complete unit starting at component development. Therefore, the design is highly subsystem-centric. Each spacecraft subsystem will have a corresponding segment on the ground

system. Each segment will provide a local server that can be integrated later into another server in a hierarchical fashion, forming the network transparency implementation. As the components are integrated, it should also be possible to integrate applications within a smaller number of segments (even going to a single segment that can handle all the spacecraft interface. The design maintains a consistent local interface from the user application to the spacecraft component, even as more ground system components are inserted between or as applications are integrated within a segment. The application and the corresponding spacecraft subsystem appear as if they were linked directly. Figures F-1 through F-3 show the evolution of this design from component development, through integration and test, and finally into operations.

In the laboratory, the local platform servers connect to a component interface that provides the logical and physical interface between the server and the components. During I&T, the component interface is replaced with a spacecraft interface that serves multiple subsystem workstations. (Depending on performance constraints, support for multiple subsystems could be integrated on a single workstation platform.) The interface between the local servers and the servers on the spacecraft interface would be identical to the interface between the local servers and the component interface, allowing applications to migrate without modification. At later stages of I&T, the interface to the ground station would be introduced. For operations, the spacecraft interface would be split into a ground station interface at the control center and the ground station itself.

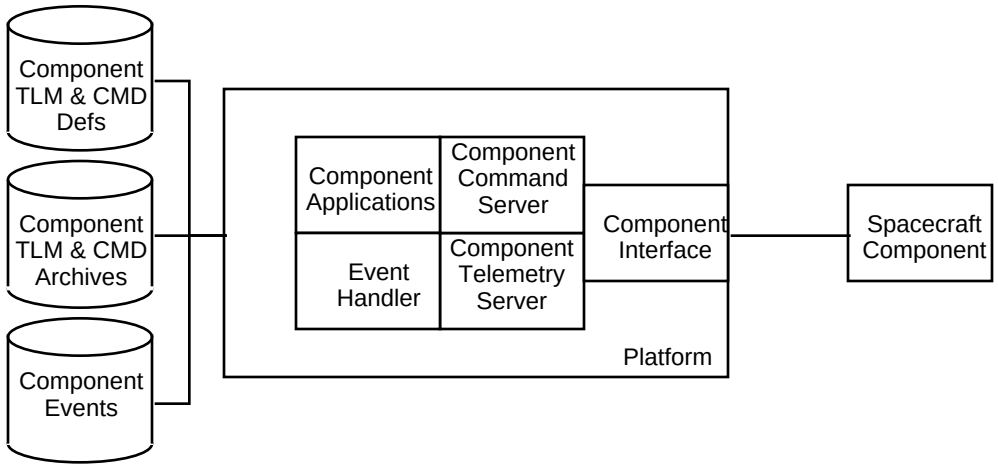


Figure F-1. Laboratory Configuration

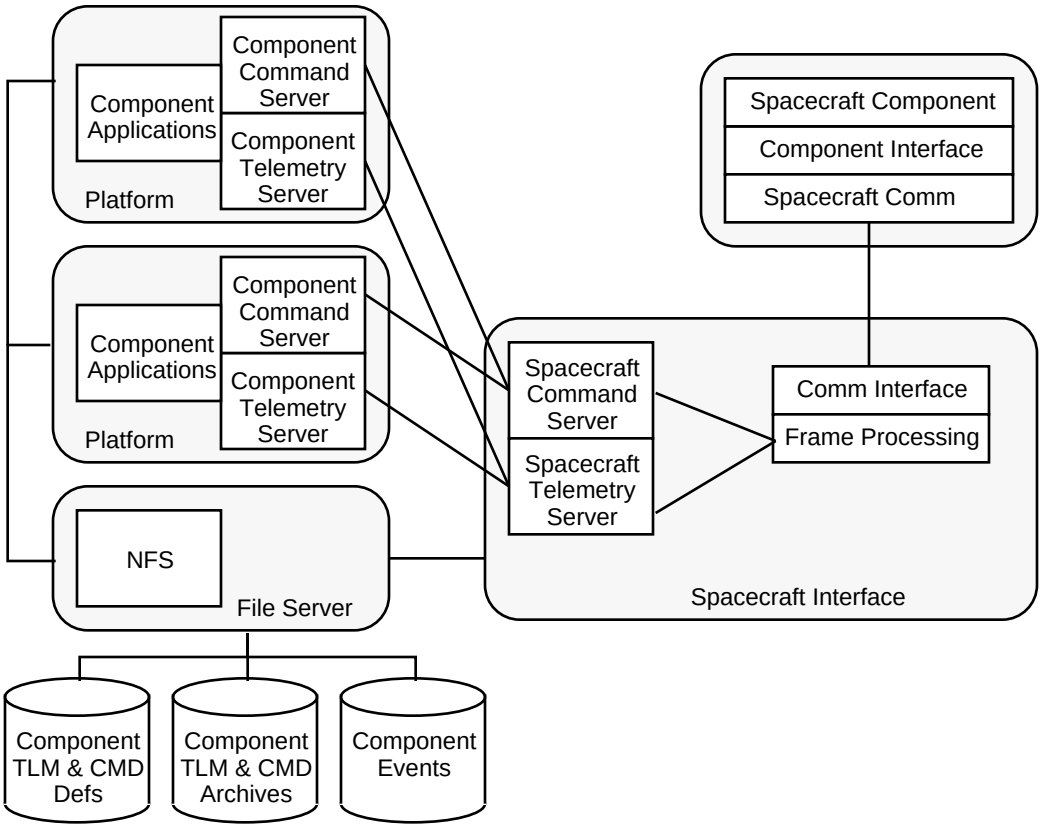


Figure F-2. I&T Configuration

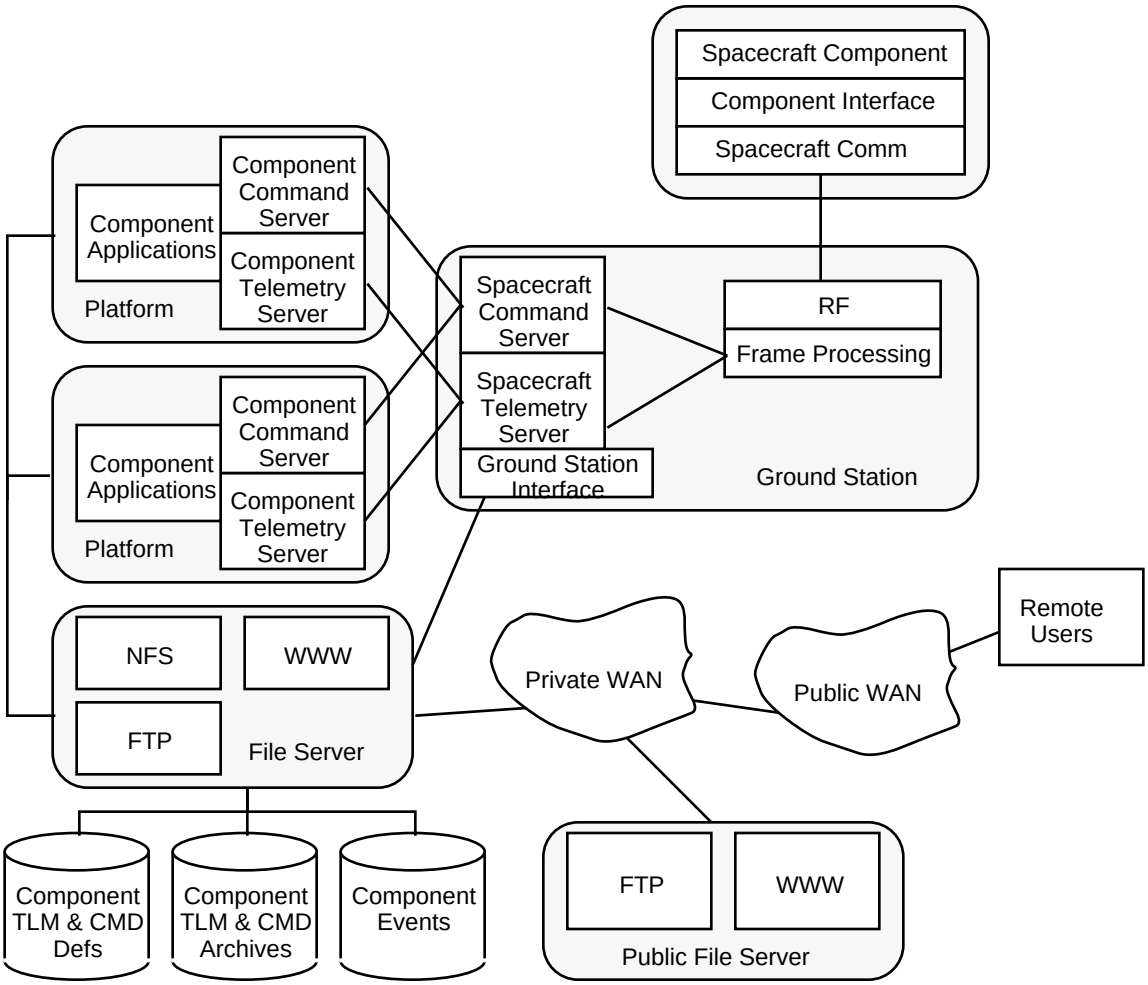


Figure F-3. Operations Configuration

F.3.2 Communication Architecture

F.3.2.1 WAN Architecture

During the laboratory phase, there is no WAN architecture. The interconnection between the spacecraft component and the component workstation is handled by an interface card, and both pieces are physically collocated. The workstation itself may be connected to the WAN for Email and other Internet access. However, this interface will be informal. Security for the WAN at this phase will most likely use the institutional capabilities at the site where the workstation is located.

During the I&T phase, the logical WAN architecture begins to take shape for the protected WAN. The spacecraft interface simulates the WAN connection through to the spacecraft. The purpose of the spacecraft interface is to hide the complexities of the spacecraft interface from the control center. If a dedicated ground station is also being developed, it can be integrated using

the spacecraft interface. At this pha[illegible]cess to the Internet will also be available. However, additional security measures wi[illegible]o be used (see Section F.4.2).

At the operations phase, the overall WAN takes shape. The protected segment is integrated via the ground station interface. Because the spacecraft interface is a logical interface that does not presume the existence of a WAN, the spacecraft interface could also be connected directly to the ground station RF equipment and the entire protected ground system located at the same site. The mission system is also connected to the public network. The mission system will provide both public access via a network server on the WAN and privileged access via a network server within the protected segment of the network. Figure F-4 shows how the WAN would appear with the spacecraft interface acting as the link between the control center and the vehicle.

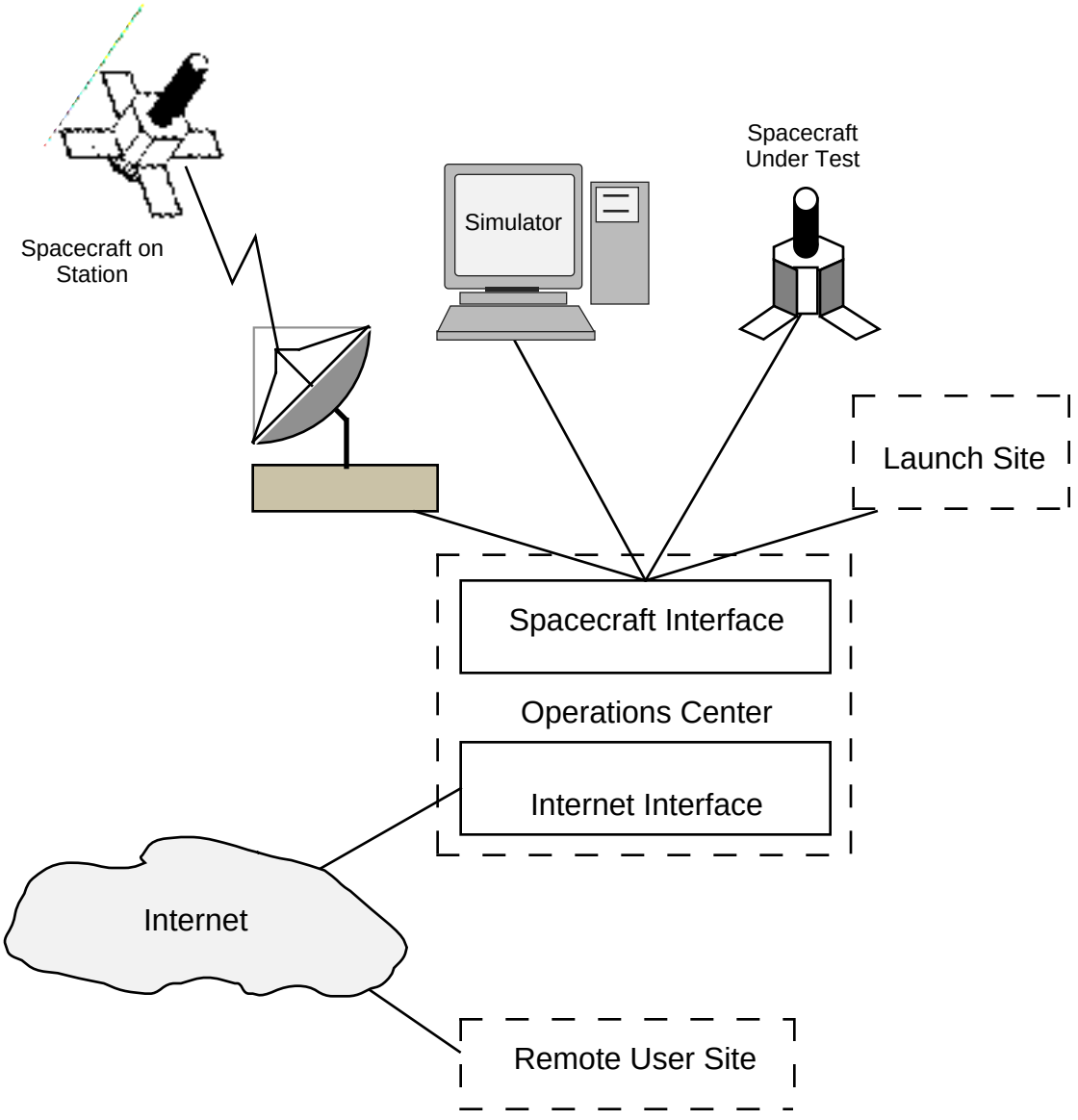


Figure F-4. WAN Architecture

F.3.2.2 LAN Architecture

During the laboratory phase, the LAN architecture comprises a single segment (as few as one workstation) connected to one or more spacecraft components via an interface device, such as the General Purpose Interface Bus (GPIB). The segment server connects to another server that communicates with and controls the interface device. Both servers are located within the segment and possibly on the same platform.

At the I&T phase, the LAN architecture begins to take shape along the lines that will be used operationally. Logically, workstations are arranged in a star fashion with the spacecraft interface as the central hub. As applications are added for operational support, workstations can either be added to the central star or added to existing segments. Physically, this architecture is well-suited to a switched Ethernet type of network. Since each point on the star is concerned with only a certain class of data, this topology would reduce the amount of traffic that is seen by each point. A segment composed of groups of workstations should be separated from the rest of the star either using another switch or a bridge. An example of how the LAN would look is in Figure F-5.

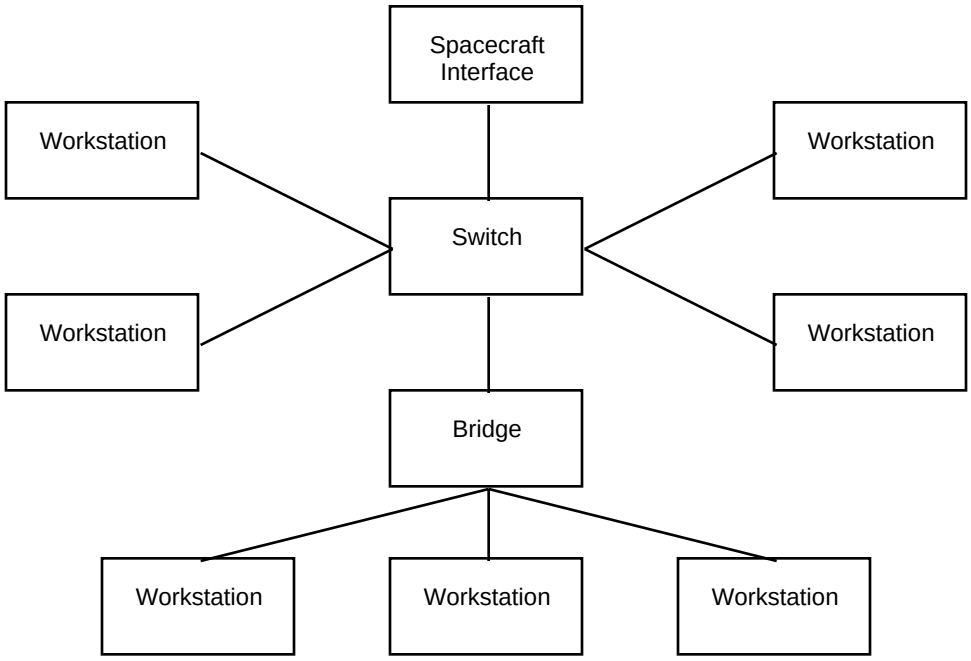


Figure F-5. Physical LAN Topology

F.3.2.3 Servers

This alternative is a hierarchical server model. Telemetry data flows from server to server to application and then on to another server (or in some cases the same server). The servers are laid out in a waterfall pattern as shown in Figure F-6. Command and control flows up the waterfall,

with each segment generating command and control information. Eventually, this information is brought together at a common point.

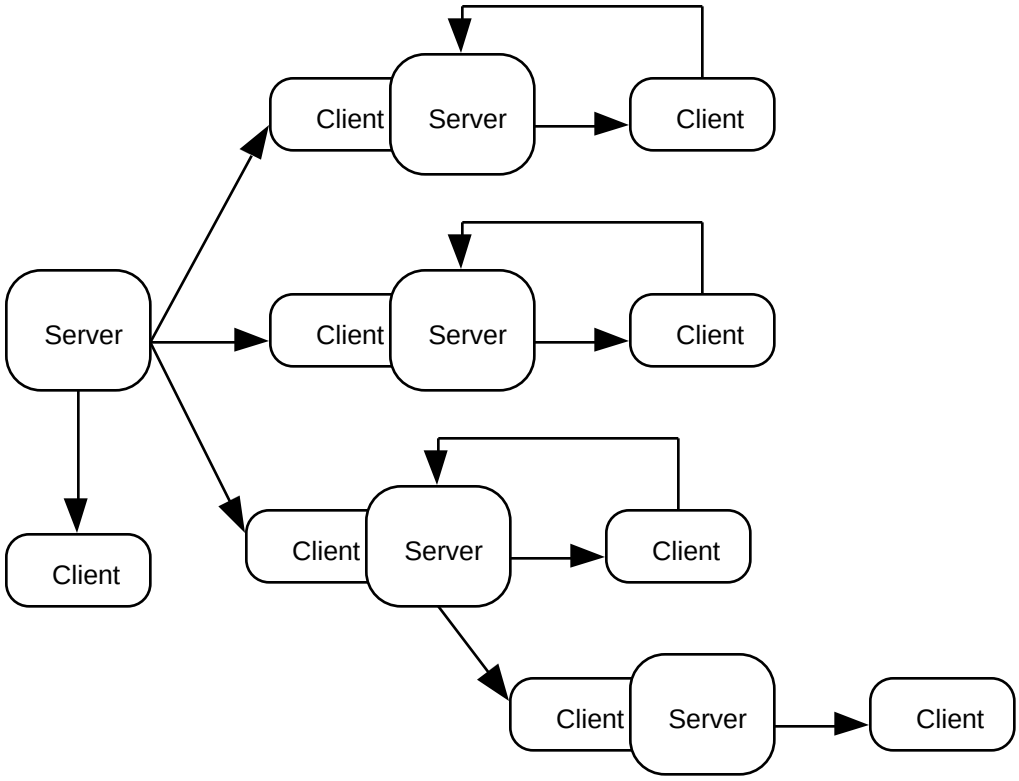


Figure F-6. Waterfall Server Model

F.3.3 Applications

Figure F-7 shows one possible implementation based on this architecture: the NASA/GSFC Code 730 ASIST system. ASIST was initially conceived of as a spacecraft integration and test (I&T) system. However, many of the concepts in the design make it extensible to the other phases of the life cycle.

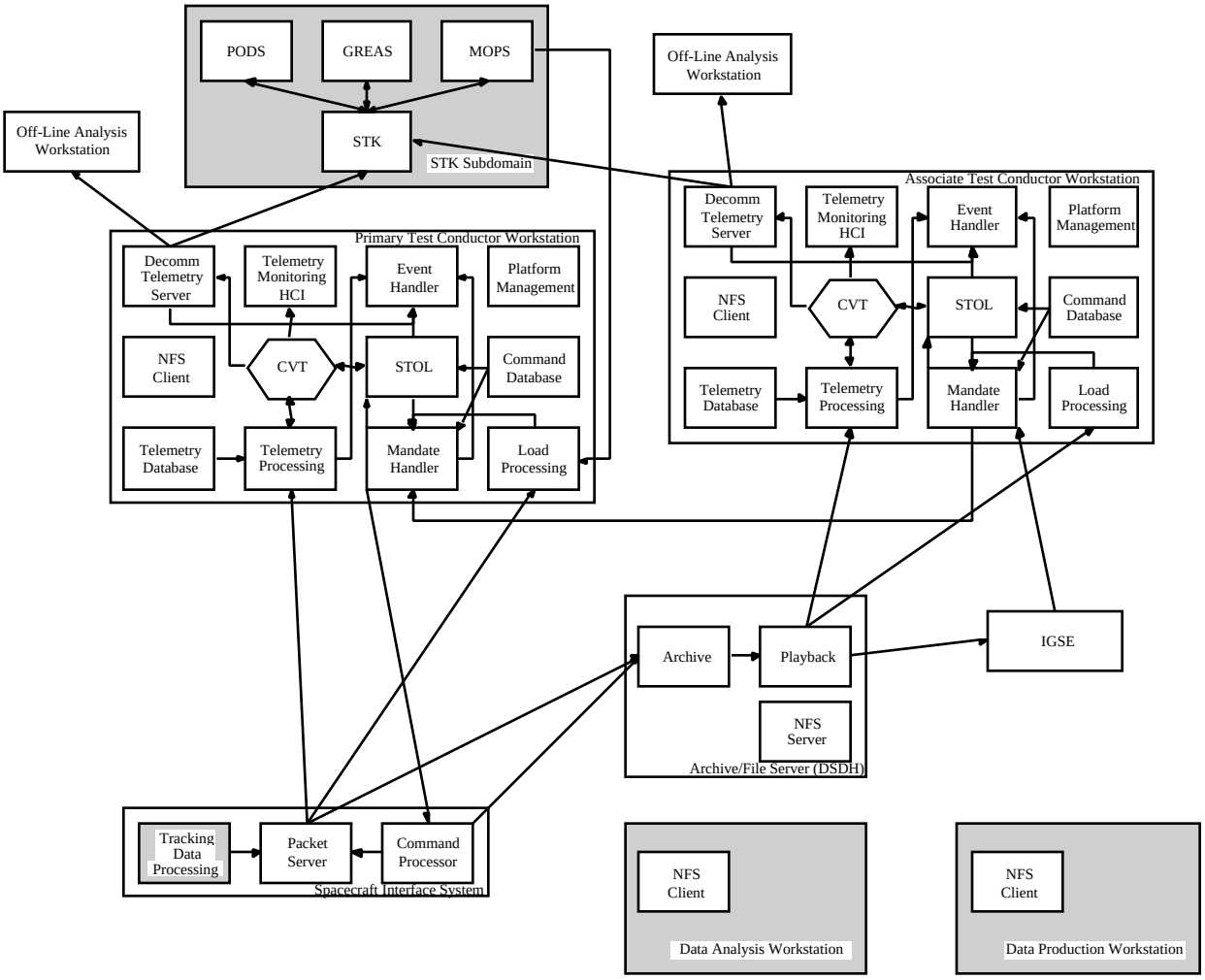


Figure F-7. Applications Allocation - ASIST

The basic concept of this architecture is to distribute the testing functions to a number of test conductors. Each test conductor is responsible for commanding and analyzing some subset of the spacecraft functionality. A common set of interfaces receive telemetry from the spacecraft and send commands to the spacecraft. The telemetry processing and displays can be customized through configuration data to process and display the telemetry in a useful format for a particular subsystem. Commanding is performed in a hierarchical fashion. Each test conductor sends commands to a primary test conductor. Command security is handled hierarchically. At each workstation, the command is filtered to make certain that it is valid, that the workstation sending the command is allowed to send that command, and that it is not a hazardous command. Hazardous commands require approval before being sent. Commands may either be for the spacecraft or for the interface system.

The ASIST distributes telemetry data through the system as SFDUs. Each SFDU contains one or more CCSDS packets. Workstations subscribe to the server to receive a particular set of packets (using Virtual Channel and Application IDs). The SFDUs are distributed to subscribing

workstations via TCP/IP sockets. Each workstation is responsible for extracting the telemetry points and converting the data to engineering units. Each workstation maintains its own current value table (CVT). The data kept in the CVT is based only on those SFDUs that the workstation has subscribed to receive. The CVT can also store data that have been created locally on the workstation (e.g., configuration information or processed telemetry points). Data security is achieved by having all requests to receive data go through the command network. As with any command, each command server checks the validity of the command against the workstation that issued it. Workstations are only allowed to see telemetry that they are authorized to receive.

The telemetry and command definitions are kept on a central file server and distributed to the workstations via NFS. Each workstation can also have its own local subset of definitions. Since the definitions are evolving during spacecraft integration, this approach allows the test conductors to modify the database during test as necessary without affecting the configured databases. This feature needs to be disabled during operations to ensure that the configured definitions are always in use.

As the ASIST system was developed primarily for I&T, there are a number of applications for analyzing and displaying data in real-time. A more limited set of applications are available for performing off-line analysis. However, no applications are available for processing attitude and orbit (especially tracking) data to generate information for mission planning. Nor are there applications for data production and dataset analysis.

The mission planning function can be met by integrating the Satellite Toolkit (STK) from Analytical Graphics, Inc. ASIST provides an API for integrating applications known as the Decommutated Telemetry Server (TSDS). TSDS provides a TCP/IP socket interface to the data kept in the workstation's current value table (CVT). Since TCP/IP can be used for both external and internal communications, the STK subdomain can either be on the test conductor workstation or on a separate platform. Glueware will need to be developed to take the information from the TSDS and make it available to the STK applications.

Data production can be integrated using the archive capability of the file server. The Digital History Data Store (DHDS) archives all telemetry received. The dataset production functions can access these data files via NFS. Once data have been sorted, subsetting functions can be performed using the tools on the test conductor and off-line analysis workstations.

The spacecraft interface system currently in use with ASIST, the Front End Data System (FEDS), is customized for the I&T environment. Telemetry is received from the spacecraft via serial (RS-422) cable. No interface is available for receiving data from the ground station, especially using TCP/IP as is currently planned for NASCOM. Nor is there a non-telemetry interface for ground station status and tracking data. This data will need to be intercepted at the interface simulator and either converted into CCSDS packets (so that existing server software can extract the useful information) or diverted to other applications that process the specific type of data. In either case, the spacecraft interface system will need to be modified to handle TCP/IP transmissions.

One significant drawback to this design is that ASIST was developed for operator-intensive operations with the spacecraft in continuous communication. No applications have been

integrated that perform detailed analysis of the data. For example, a finite state machine (such as Altair MCS) could be used to determine if the spacecraft is in a good state (when used in combination with a spacecraft simulator, it can be used to determine the impact of a certain set of commands). Implementing the Altair MCS would require either that the MCS be modified to read data from either the TSDS or the CVT, that an interface be developed between the RTserver portion of the MCS and the CVT, or that the CVT be replaced with the Talarian RTserver. The approach used would depend on the amount of code that needs to be written or modified and the number of RTservers that would need to be purchased. Since any of these approaches requires that the spacecraft state be predicted during non-contact periods, this function could also be moved to the spacecraft. The spacecraft state machine could then check the impact of commands based on the actual state of the spacecraft.

With the exception of the PTCW, all telemetry is routed through and filtered by the Digital History Data Storage system (DHDS). (Theoretically, all telemetry could be sent from the FEDS to all TCWs, but the system is unable to support this operationally due to performance constraints.) The DHDS archives the telemetry as it is received. The data is then played back to requesting workstations, with a potential delay of 15-20 seconds between FEDS data packet receipt and receipt at the workstation. This approach simplifies the interface for playback but creates problems for operational systems. First, the server itself is primitive and was developed in-house. As a result, it uses a custom protocol for interfacing with the workstations (based on TCP/IP sockets). A more effective approach would be to replace this server with a commercially available server and place resources on the problems that really need to be solved. Secondly, the delay through the archival system is acceptable for I&T but not acceptable for real-time operations. Again, a commercial server would be more useful, especially one such as Talarian RTserver, which includes its own archive and playback clients that operate independently of the server process (that is the serving of data does not rely on the archive process and playback mechanism).

The local segment servers in the ASIST implementation are implemented as the current value table (CVT). The CVT is a shared memory server, requiring that applications all reside on the same platform. While the decommutated telemetry server is available for connecting applications from other workstations, applications must be written with one or the other approach. Most of the existing ASIST applications were written to use the shared memory approach, making it difficult to move the applications to other workstations without replicating the entire ASIST system.

The combination of custom servers appears to have limited the choice of implementation platform for ASIST to UNIX and according to the ASIST users and documentation may limit it to IBM RS-6000 workstations. This limitation makes it very difficult to scale ASIST to the level that may be needed by a laboratory developer (e.g., an instrument developer may wish to use a PC), thus making it nearly impossible to push the architecture down to earlier phases of the mission lifecycle.

Commanding is resource-intensive in that it requires that the command pass through the PTCW. This requires at least a two-step transfer of data across the network, without modification. At the least, the final mandate check could be performed at the interface system. As already discussed,

it should be possible to move the check to the spacecraft. If commanding is to be done from a remote location, a security server, such as Kerberos, should be used to authenticate the source of the command. When used in conjunction with encryption, Kerberos should be able to prevent someone from masquerading as a legitimate user and sending commands to the spacecraft. Also, the approval of hazardous commands is also operator-intensive. Whenever a command is sent to the spacecraft, it is filtered at the PTCW. If the command is deemed to be hazardous, then an operator must review the command. The use of a finite state machine could be used to determine the impact of each command and determine if it will adversely affect the spacecraft without human intervention.

The commanding scenarios used in I&T are much different than those used in operations. In I&T, the commands are sent directly to the spacecraft (after having been validated by the mandate handler) and executed. This path will work fine for real-time commanding during operations but not for command uploads. The system will need to have two command servers, one which receives commands to be uplinked directly to the spacecraft and the other which receives requests to schedule events on the spacecraft. For example, if monitoring the tracking data indicates that an orbit maneuver is required, a command would be sent out from the application that monitors the orbit. Obviously, this command should not be uplinked directly to the spacecraft. Instead, it would be sent to the mission planning system as a request to change the spacecraft orbit to the required set of orbital elements. The mission planning system would schedule the event (including making the necessary calculations and creating the spacecraft commands to execute the event).

F.4 Analysis of Design

This section describes the approaches used in this architecture to enhance the technical performance of the system.

F.4.1 System Performance

The segmentation of the design within the operations center provides several opportunities for tuning the overall system performance. Since each segment is designated to perform processing for a particular component or subsystem, the allocation of hardware and processing power can be tuned for that particular segment. Also, since each segment works on only a particular data stream from the spacecraft, the network can be segmented so that each segment only receives the data that it will process (and no other data even appears on the segment). This reduces the amount of network traffic overhead. Furthermore, in the ASIST implementation, the databases can be distributed so that each segment could have that portion of the database that is unique to its processing. Only those parts of the database that are common to two or more segments would need to kept on the file server.

F.4.2 System Security

Because of the distributed command features of this design, a method is needed to guarantee that commands are being sent as authorized. These can be spacecraft commands, system commands to direct telemetry to a specific node, or commands to control the ground station. In the ASIST

implementation, each workstation has a mandate handler. The job of the mandate handler is to verify that each command has been sent by a valid source. Commands are passed up a chain of mandate handlers until they reach the final destination (which may be an application on the originating workstation or the spacecraft itself). The mandate handler verifies that the workstation sending the command is authorized to send that command and, if the command is deemed hazardous, alerts an operator who must verify the command. This security could be augmented through the use of Kerberos to validate the workstation and a state machine that could be modeling the state of the system to verify that a combination of hazardous commands does not threaten the mission system.

In addition to command security, some telemetry may need to be secured. This is particularly true of proprietary science data. Encryption could be used on the telemetry stream from the spacecraft interface system onward to protect the data.

External login security to the system can be controlled via a firewall located between the file server and the external network. Since all real-time functions will be performed within the control center, access from outside can be limited to file transfers with the protected file servers. Transfer of data into the mission system would need to be controlled through a filter that would only transfer data from known sources.

Internal login security can be controlled through the use of smart card technology. The combination of firewall and smart card technology should adequately protect the system from unauthorized access.

F.4.3 Availability

This architecture has three features that affect the availability of the mission system. First, in the control center, the availability needs to be engineered at a segment level. Since the largest part of the availability question arises around the ability of the segment to receive telemetry and send commands, the need for a particular segment during a pass needs to be analyzed. If the analysis functions can be carried out without direct contact with the spacecraft (i.e., real-time reaction to telemetry is not required) and the telemetry can be captured without the segment, the segment does not need to be online during every pass. The availability of the segment only needs to be match the needs for completing the analysis in a timely manner. If, on the other hand, the functionality is required during the pass, some ability needs to be available to move the functionality to another segment. The ASIST implementation ties commanding to workstations. Therefore, the configuration would have to be updated to move the functionality to another segment. On the other hand, the implementation could also be modified to tie commanding to user or to some process identifier. The security risks of such a change would need to be weighed against the need to be able to quickly reconfigure the system.

Two potential single points of failure affect the availability of the mission system. The first single point of failure is the combination of the primary test conductor workstation (PTCW) and the spacecraft interface. This combination receives all telemetry and forwards all commands. In the ASIST implementation, these connections rely on socket communications to known addresses. During I&T, this approach is satisfactory, since the time to restore would not adversely affect the I&T. However, during operations using such an approach would take too

long to reset the configuration. Therefore, as mentioned earlier, the socket communications should be replaced with a commercial server that can be replicated on more than one platform. The Talarian RTserver is a good example of a server that would fit this situation. The second single point of failure is the file server. Since this server is used to store the command and telemetry definitions, it needs to be available almost continuously. This server should be configured as a redundant system. This server could be implemented using a fault-tolerant system such as Stratus or using fault-tolerant server software that switches between two or more server platforms, such as Sun's High Availability option. In either case, the data kept on the file server must be safeguarded from failures. RAID is one technology that could be used. If the online volume of data is too high for RAID, optical disk jukeboxes have many features that enhance the availability of the data (not to mention enhancing the completeness of the dataset).

The final feature of the system that affects availability is the use of multiple command sources. An unfortunate combination of commands could place the spacecraft in an unsafe mode, thus disrupting the use of the system and endangering the mission. At best, the spacecraft would go into safehold, waiting for instructions to rectify the situation and losing several hours of operations time. As mentioned earlier, the use of a state machine to monitor the state of the spacecraft (or expected state) would greatly reduce the accidental loss of the spacecraft through unfortunate combinations of commands. This state machine could run on the ground and monitor the expected state of the spacecraft after each command or it could be a part of the spacecraft and check the commands against the actual state of the spacecraft. In the second case, alternative command sequences may need to be made available to the spacecraft to keep it from becoming nonproductive.

F.4.4 Data Quality

This design offers little innovation in data quality but provides few weaknesses that degrade data quality. The quality of the spacelink is completely up to the design of the spacecraft. Reed-Solomon encoding would be expected to be used for this link. The data archiving of the ASIST is the strongest point for data quality in the control center. Since data can be archived at each workstation as well as at the archive subsystem, the completeness of the dataset and the availability of data for analysis are practically guaranteed. The DHDS implementation uses a system that automatically archives data to optical disk as the online disk storage becomes full.

Appendix G. NASA-Integrated Mission Design

G.1 NASA-Integrated Mission Design

This sample design provides a NASA-integrated mission design based on use of STI OS/Comet as an integration system throughout the mission life cycle. This product provides a foundation command and control capability appropriate for use during spacecraft I&T, with extensions included for use during a mission operations phase. It can be hosted on as little as a single workstation with hardware interfaces and extended to an arbitrarily large number of operational work positions. The sample design is based on currently available applications and infrastructure components.

G.2 Operations Concept

The mission concept includes integration and test, followed by evolution of the ground data system to operations status in support of Launch and Early Orbit activities and nominal operations. The assumed mission is integrated with NASA resources and uses technology similar to that routinely used for current missions. Standard telemetry is assumed to be the interface to the spacecraft, with a single ground station for operations in the model of SMEX-lite.

During I&T, the integration will begin with test of the spacecraft data and control interface components. Following validation of the interfaces, the spacecraft bus subsystems are added one by one for integration tests. Finally, the payload is tested with the rest of the spacecraft. It is expected that standard test equipment, e.g., signal analyzers, oscilloscopes, etc., will be used to validate the signals before any component is attached to the automated analysis and test system. During mission integration tests, NASA communications and launch support will be used to interconnect NASA resources to validate all operational interfaces.

During L&EO preparation, the I&T system may be also used to validate the integration and checkout of the launch vehicle (if commercial vehicle). OS/Comet provides a standard mechanism to address the entire commanding and interpretation configuration as self-contained objects, providing validated isolation between sets. The I&T system will be used to validate payload operations during the prelaunch activities and to perform the unique activation processes for the early orbit operations. NASA launch support is assumed to include the launch facility and one tracking station for launch vehicle tracking data collection.

For the operations phase, a copy of the ground data system is instantiated at a customer site for customer control of spacecraft operations, including both bus and payload. Preliminary payload analysis and product preparation are co-located with the spacecraft operations to form an integrated operational entity. Final product analysis is assumed to occur at a number of remotely located customer sites, which may or may not be on the same campus as the operations center. Transition to the operations phase instantiation is assumed to occur between payload I&T and L&EO phases.

G.3 System Design

The system design provides the overall design for this system, with evolution of features shown through the mission life cycle. The section is structured to show the overview of the design, followed by more detailed definition of the communications, software integration, and application designs.

G.3.1 System Level Design

The general design of the OS/Comet core of this system is shown in Figure G-1 below. This figure illustrates the central role of the Software Bus in acting as a network transparency implementation to applications within this design. The Software Bus acts so as to make transparent the location of the data sources and the applications, providing a common interface independent of location.

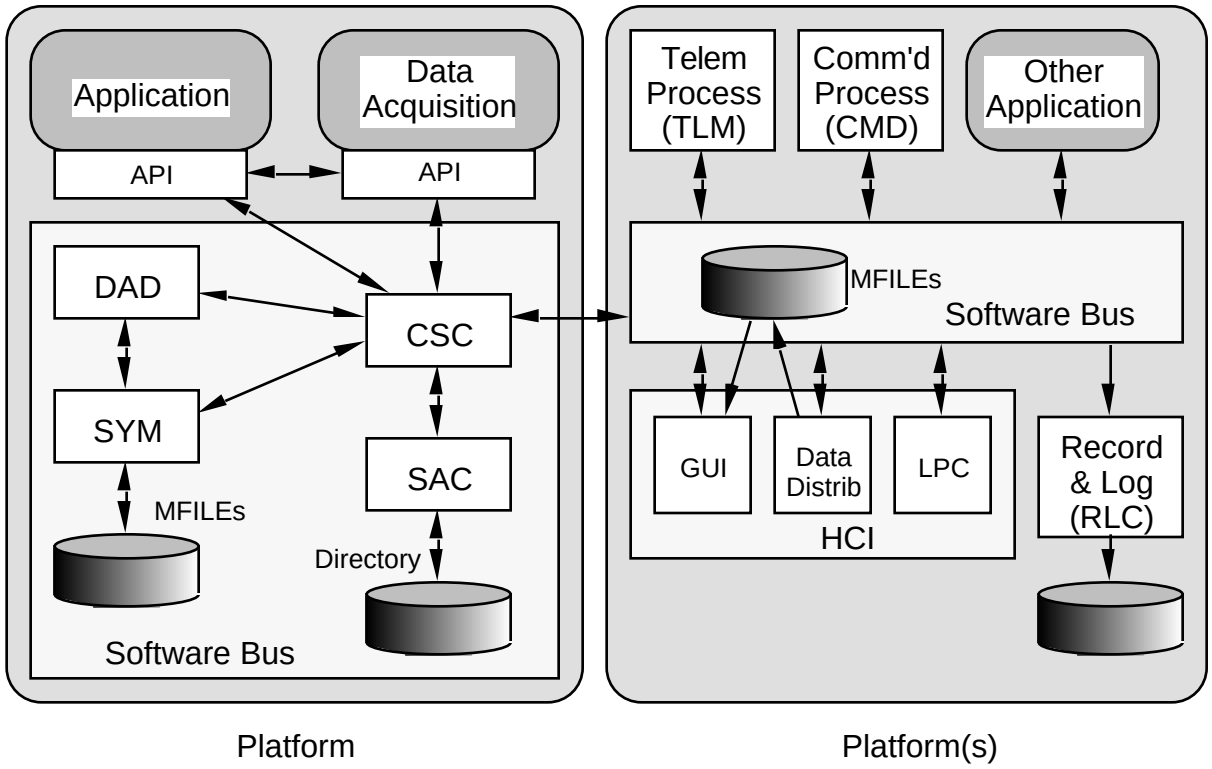


Figure G-1. OS/COMET Core Design

The figure shows a sample of the interfaces for the standard applications provided by OS/COMET as the baseline product. Note that Data Acquisition, Record & Log, Telemetry Processing and Command Processing are standard applications. With this design, the choice of the number of workstations for I&T activities is determined by the operational approach. Either single workstations can be configured to operate independently or a small set can be configured to operate in cooperation for a given test configuration.

Figure G-2 shows a configuration for early test operations with a single workstation interfaced to an instrumentation component through a hardware interface. Depending on the interface standard, this interface may be either a board in the workstation or a stand-alone electronic interface. The interface can be changed to match any of a variety of standard Command and Control (C&C) drivers included in OS/COMET or may be custom adapted in the Data Acquisition module. The workstation is entirely self-contained, requiring no integration with other test system elements.

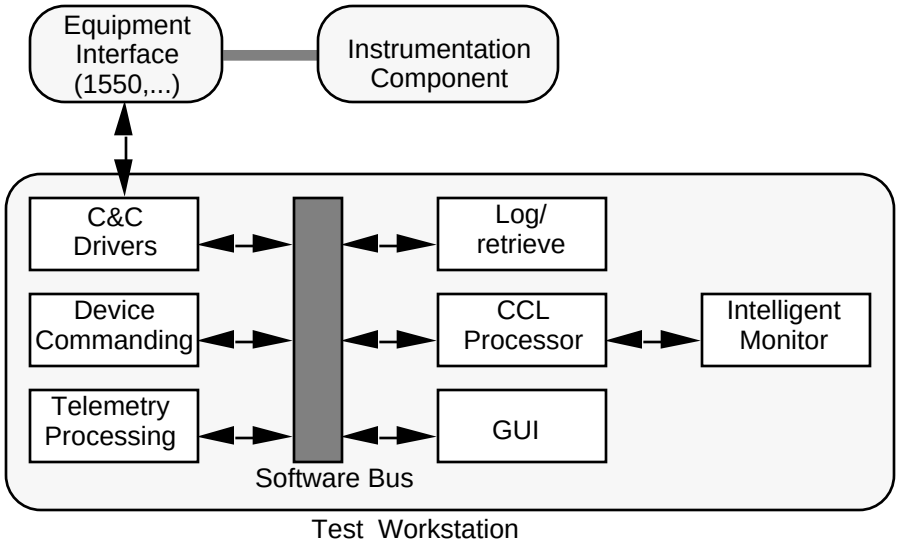


Figure G-2. Single Workstation Test Configuration

When the integration testing reaches the stage of cooperative testing of components, e.g., thermal/vacuum tests, then the configuration can be modified as shown in Figure G-3 to include multiple workstations in support of the test.

In this figure, there are four platforms, one communications processing hardware interface, the test support equipment and the spacecraft being tested. There is one platform for command and control processing for the tested spacecraft, one for test equipment control, and two analysis support workstations for data analysis of subsystems as needed.

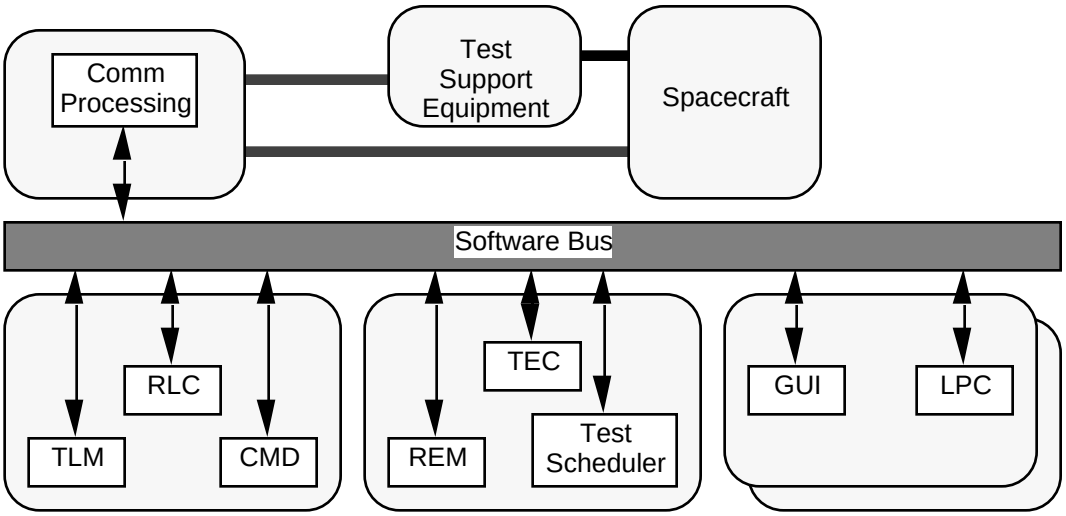


Figure G-3. Integration Test Network Configuration

In Figure G-4, the operational configuration is illustrated. The design has been extended to include support for the overall ground system, orbit determination and prediction, and product generation. The STK application subdomain is included based on the assumption that tracking data are received and converted to orbit elements. GREAS provides predictions for orbit-related events. Other members of this family, e.g., Navigator, may be added where appropriate, on this platform and attaching directly to STK. RTie is added to present a capability for intelligent automation of the operations.

It should be noted that the allocation of processes to platforms represents only an initial set, and that OS/COMET allows porting of these among the platforms to best suit operations and performance needs. There are additional capabilities not shown here, for segmentation of the network and for extended availability.

For connection to the public Internet, the communications processor must interface to a security gateway platform that isolates the above components from public access. Additionally, the Software Bus would be configured to incorporate DCE into the its access and interfaces.

G.3.2 Communications Architecture

The communications architecture for this system is fairly simple because it is operationally an autonomous mission. Only during the early phases are NASA facilities used. For launch support, the communications process provides an interface to the remote NASA site and a network connection to NASA sources via a NASA maintained connection. These comprise almost the entire wide area support, and are all removed shortly after launch and not used again. The only other element is for product distribution to remote customers. This is provided through the security gateway introduced in the previous section.

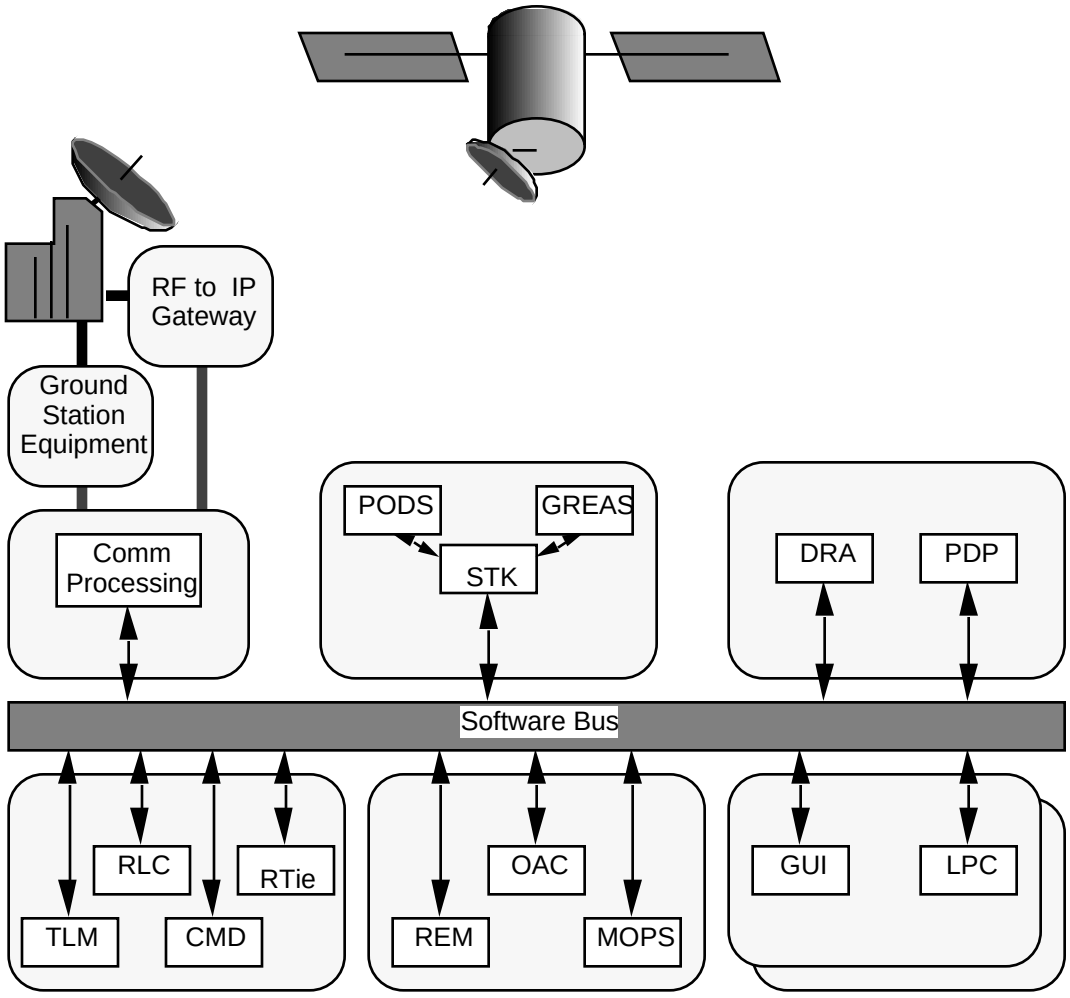


Figure G-4. Operational Configuration

Given that the spacecraft is NASA built and that telemetry is the primary data exchange mode, both IP and CCSDS data formats will be in use. The RF to IP gateway shown in Figure G-4 would be expected to produce IP packets from CCSDS formatted telemetry, and provide CCSDS formatted commands to the spacecraft, based on data received over IP from ground sources. CCSDS traffic would be limited to the RF path between the spacecraft and the ground.

G.3.3 Software Integration Architecture

Figure G-1 shows the heart of the software integration architecture, the software bus, combining network transparency and communications. Within the local area, this software bus provides the integration architecture for most applications. As shown, the software bus is based on an architecture of replicated servers, with equivalent servers on every platform served by the bus. The components of the bus are: the Communications Services Component (CSC), the System

Administration Component (SAC), and the Data Distribution (DAD) and Symbol Processing (SYM) components.

The CSC provides the primary mechanism for data interchange. It provides the interface (API) to all applications and performs the platform to platform message transport process (over IP). Messages can be multicast in a variety of forms, depending on the details of the configuration, with the goal of performance optimization.

The SAC implements a distributed name service which may be implemented through DCE. It performs process initiation based on configuration file entries and/or run-time requests. It also monitors the process state to determine process unavailability and trigger reconfiguration when needed.

The DAD process manages the replication of data across platforms. For each service table (MFILE), the DAD tracks the updates, and the processes using that table. Messages are generated to update the MFILE on each platform with a registered user, using the CSC for actual message transport.

The SYM component provides the basic mechanism for entering and sharing data via the MFILEs. An off-line facility processes database definitions into runtime structures. The MFILEs are essentially tables of symbols, each with current value and attributes. Applications use SYM functions to read and write data to the MFILEs. SYM processing can be used to generate formatted alarm messages when data are changed to an abnormal state or value. SYM can also trigger application processing based on value updates.

Additionally, the STK interface provides a common file exchange interface for those applications in its subdomain. In this version of the design, MOPS is used as the mission planning application and interfaced to the software bus. An alternative is to attach it to STK and use that mechanism for generating the details. The bus seemed the better design for this because of the closer integration achievable with spacecraft and ground system commanding.

G.3.4 Applications

The applications identified in this design are a mixture of COTS and other forms of off-the-shelf products, with minimal development. In some cases, the availability of off-the-shelf software applications depends on other choices. For example, there are a limited set of communications interfaces and ground antenna control systems for which there are existing applications. A choice for a different implementation can be made, as long as the cost and risk impacts are assessed and accepted.

Table G-1 shows the list of applications as illustrated in Figure G-4, with further identification as to use of COTS or other OTS products.

Table G-1. COTS Application Allocations

Component	COTS Application
Spacecraft Control	OS/COMET CCL, Talarian RTie
S/C State Monitor	OS/COMET CCL, Talarian RTie
System Management	Hewlett Packard OpenView
State Analysis	Talarian RTie
Operations Flow Director	OS/COMET CCL
State Prediction	Orbit: PODS; Trending: OS/COMET HCI, RLC; BBN Probe
Mission Planning	MOPS
Product Generation	TBD
Product Analysis	TBD
Communications	Omega; LTIS 540
GS Control	OS/COMET OAC; MMS application (?)
GS State Monitor	OS/COMET REM
Security Gateway	High end options: Harris CyberGuard Firewall, Digital Firewall for UNIX

The only area left open above are the product generation and analysis applications. While there exist products such as Pacor II and the PPS which may be applicable, these are poor fits to the architecture and may be more costly to use than off-the-shelf components in the OS/COMET venue or elsewhere. For this reason the selection is left open at this time.

G.4 Analysis of Design

G.4.1 Benefits

The following are the leading benefits of a design based on OS/COMET:

- High capability, common API at the LAN level
- Extensions to the bus interface to more global standards: CORBA, DCE
- Built-in scripting with direct links to intelligence and multi-threaded activities
- Designed for both I&T activities and operations automation
- Inherently designed to manage multiple objects: spacecraft, ground stations, etc.
- Mature technology

The software bus is a high capability design, providing network transparency and a high level of functionality for applications using it. The combination of SYM and DAD provide a very flexible approach to provision of data in the desired format with control linkage possible. This

provides a single, comprehensive API that all applications can use, with connections to system management and configuration databases. Also, STI has provided an extension of this API to a CORBA/DCE interface to support an extended set of COTS applications.

The internal scripting capability (CCL) provides for a general, multithreaded approach to activity planning and execution. This supports simultaneous and coordinated operations for linked, but independent, elements such as the spacecraft and the ground antenna. This is also linked to an object approach to component configuration that supports simultaneous operations with multiple spacecraft (not used here). Finally, CCL is a translated language, supporting the needed flexibility for I&T activities, and with links to automation support capabilities such as the RTie inference engine.

OS/COMET has a proven track record and uses mature technologies to reduce the risk of use. There is a true engineering capability behind it, providing critical support over the long run. Additionally, it has attracted a reasonable suite of third-party vendors who have created interfaces to the software bus for their products.

For the specifics of this mission configuration, there are related benefits:

- Common system migration through the mission life cycle
- Integrated end to end operations
- Extensive use of COTS applications

Using the flexibility of the base product allows for use of a common command and control base throughout I&T into operations. This ensures optimal use of I&T time to validate configuration data and develop operations rules for later automation. Depending on the contractual relationship between the end customer and NASA, licenses and hardware from I&T could be delivered as a part of the operational system.

The design uses the capabilities of OS/COMET to provide for integration of the ground system operations and product generation into one LAN environment. The robust capabilities of the bus provide the flexibility to support this.

G.4.2 Risk and Mitigation

There are two primary risks associated with this design: a proprietary API, and potential performance bottlenecks in data transport.

Most real-time APIs are proprietary today. The lack of high-level standards is the risk with most impact potential. A failure of the vendor or even a major shift in the API could lead to a large expense to adopt a different standard API. This risk is not huge because the vendor is not too small and weak, and for a given mission, one may be able to live with an unsupported product for the remaining life. Risk is also mitigated by minimizing development around the API.

The replicated data and server architecture provides high responsiveness to applications acquiring data from the local servers MFILEs. It carries the potential problem of large volume data transport to multiple platforms if updates are not properly managed. Historically, the limits

of the original OS/COMET design were visible on Clementine, where image data were transported across the network. Since then, a number of improvements have made the limits much higher. Potential use of FDDI removes one of the biggest problems, the original restriction to Ethernet because of the use of Ethernet broadcast for distributing data. The LAN can also be segmented to avoid having all data go over the same "wire," thus removing some of the peak features. Further mitigation will be undertaken by prototyping mission operations with this product to assess whether the issue is real for the given mission before committing to the design.

G.4.3 Operability

The operability of this approach is quite high, in that it combines many desirable features. The key elements are

- Flexibility of database and CCL for I&T
- Ease of overlaying CCL with automation
- Configuration flexibility to support automated failover and recovery

The origin of OS/COMET in the I&T environment shows in its inherent flexibility for configuration. The scripting language, the database, and the generic interface to device drivers create an environment highly adaptable to changes resulting from unexpected behavior, and supporting ready repetition of events for diagnostics. Since the configuration can be built with only the needed elements, the light version can match the needs of any given test.

The progression to operations shows in the ability to add automation. Originally, CCL scripts were considered sufficient, and have a high level ability to deal with simple conditionals on the environment. With the recognition of spacecraft operations complexity, STI teamed with Talarian to develop a standard interface to RTie as a complement to the basic CCL capability. This interface can be used to extend over RTie using Altair's MCS, though this brings in significant redundant capability at this date.

The software bus naming service supports dynamic observation of active processes and reconfiguration to a backup as a recovery mode. This basic capability provides the ability to meet most mission requirements for availability without expensive duplication of all hardware. It also provides an automated switch over, avoiding the need for human intervention and error.

In conclusion, OS/COMET provides a proven operations environment meeting the needs of most missions. For missions depending on real-time interaction with the instrument and spacecraft, it provides a consistent approach to full mission operations. It has a long record of operations with multispacecraft systems in DoD and near-term experience with other customers. The operations rough spots have been thoroughly sanded off.

Appendix H. Technical Performance Approaches

This section provides an analysis of the approaches to technical performance that are intended to result from the architecture in the preceding sections. Technical performance includes the following four categories of engineering performance:

- System Availability
- System Performance
- System Security
- Data Quality

Rather than restrict the discussion to the general architecture, two types of mission systems are used to illustrate the approaches resulting from the architecture. The first is a system designed within the context of current standard operational procedures, in which the mission system is closely integrated within the NASA context, making heavy use of existing NASA resources. The second is designed as a much more self-contained system, assuming autonomous mission operations, i.e., little or no NASA involvement with operations postlaunch, and responding to a single coherent customer.

The design for the NASA-integrated mission system is illustrated in the interconnection diagram of Figure H-1. The assumed mission approach includes all the pre-operational facilities within the NASA context, interconnected through the protected NASA network. During the operational phase, the end customers are presumed to be located at more than one place within NASA, and external to the NASA environment. For some, most communications are through NASA organized networks, either internal or leased line. Others connect only through the Internet. The mission network forms a Virtual Private Network, running over several providers, with some segments NASA administered, and some public.

The mission example assumes an operations facility located at Goddard Space Flight Center, connected onto the protected NASA MODNET. This campus network serves customers across the GSFC location and has wide area connectivity to remote NASA and other government locations. Access to this network is controlled by NASA, and the network provides a security gateway connecting a side with public Internet connections to a protected side with only selected government nodes. Inside the protected MODNET, there are only basic security capabilities for most facilities, with the presumption of isolation to support mission security.

During operations, data flows between the spacecraft and the operations center via the Space-Ground network and the protected MODNET. Customers exchange planning data and commands to the spacecraft through the operations centers at GSFC, using the variety of network connectivity illustrated. Similarly, payload data products are distributed to the customers over that same mix of communications networks.

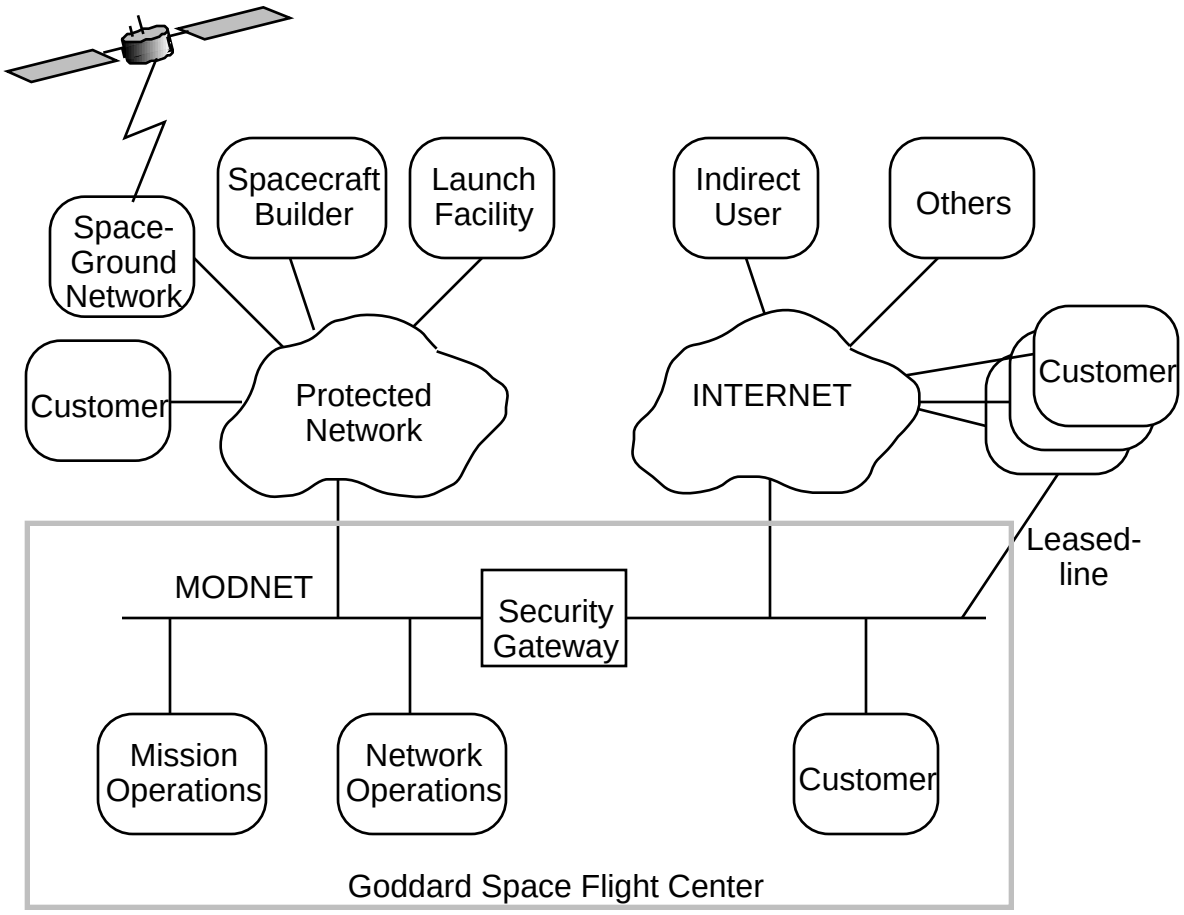


Figure H-1. NASA-Integrated Mission System

The autonomous mission system is illustrated in Figure H-2. The mission concept is that all elements of the mission system are not integrated within NASA, though some, e.g., the launch facility, may be a NASA site. The operational ground station and mission operations facility are assumed to be collocated and also one data customer, as in the SMEX-lite concept. Other customers, e.g., and instrument team may be located elsewhere, and customers of mission products are assumed to remotely distributed over the Internet, as in the EOS model.

Again, the intent is to form a virtual private network connecting the spacecraft, ground station, mission operations center and customers for the operations phase. The launch site interconnects to the operations center during launch checkout and returns launch vectors for early orbit determination at the operations center.

Commands and operational data flow between the spacecraft and the operations center through the ground station and the local network. Planning and commands may flow locally or over the Internet from customers. Similarly, data products are distributed from the operations center to the data customers either locally or over the Internet.

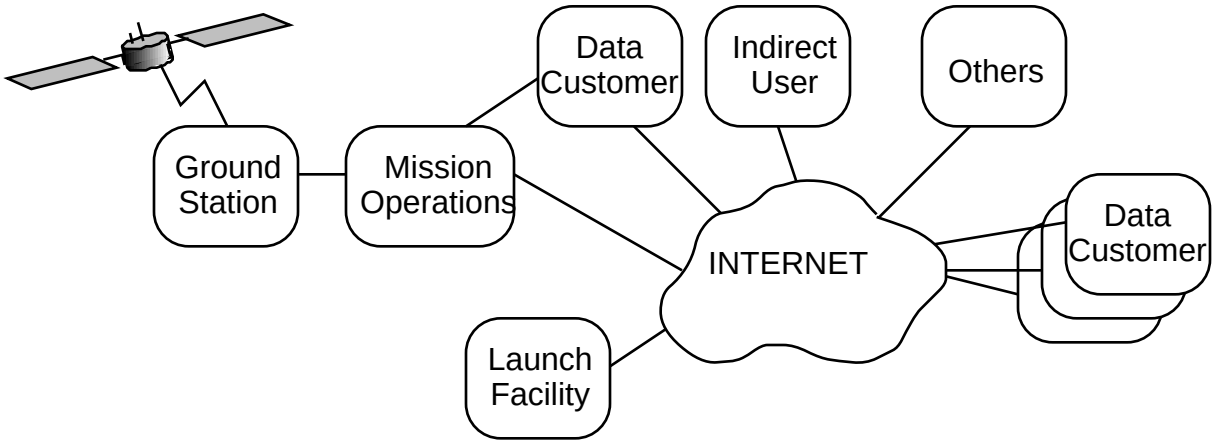


Figure H-2. Autonomous Mission System

H.1 Mission System Availability

System availability is the most common of mission system requirements. It is an attempt to address the basic ability of the system to perform those operations needed for a successful mission. System availability is a metric that becomes complicated to define in real life because mission operations are not constant in mode and in resources used.

For the Renaissance architecture, the system is a distributed network of those components needed to perform the tasks of the mission, such as shown in Figure H-3. In such a network, all the components and connections needed for the specific operation must perform properly when needed. This is made more complex by the fact that only some components are needed for any given operation, as indicated by the shaded blocks in the diagram.

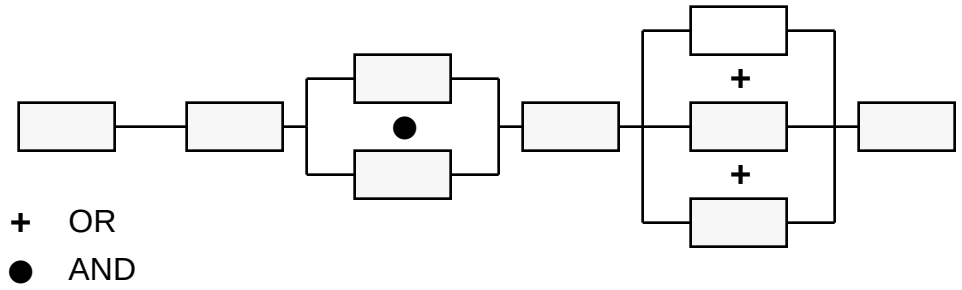


Figure H-3. Activity Component Use

Further, the system will generally be configured in such a way that some parallel components are both needed (indicated by AND), and others in which only some are needed (indicated by OR). It should be noted that in a real network diagram, hardware, software and data components are equally regarded and needed for overall availability. It is not sufficient to consider just hardware.

The three sections below describe the availability requirements as typically apply to space missions, the strategies and tools that can be applied to meeting such requirements, and overall alternatives using the mission systems of Figures H-1 and H-2 as illustrations.

H.1.1 Availability Requirements

System availability is defined to be the fraction of the time that the system capabilities are available. Mathematically, availability is related to two other metrics for calculation by the equation:

Availability = 1 - <MTTR> / <MTBF>

where <MTBF> is the mean value of the time between system level failures

and <MTTR> is the mean value of the time required to restore system capability given a failure.

It is important to note that both MTTR and MTBF refer to system capabilities, not necessarily component failures. Further, note that MTTR is the time it takes to restore capability, not the time it takes to repair a failed component.

In specifying system availability, it is common to specify more than just the availability fraction, and in fact other related metrics may be used because of the complexities discussed in the introduction. Typically, either MTBF or MTTR will be separately constrained because this is the most critical characteristic. For example, a mission that depends on on-line commanding of the instruments may specify a maximum MTTR as a way of ensuring there will be no significant down times. A mission wishing to ensure minimal interruptions in operations may specify a minimum MTBF.

A different way of specifying availability is to provide the requirement, with an explicit link to the activities that are crucial to the mission. For example, systems with critical control operations may wish to specify a high availability for command and control activities and lower availability for other areas to reduce costs.

Today, mission availability is usually separately specified for different components of the system, e.g., the spacecraft and the ground data system. This has its origins in the routine use of shared facilities in which there was little chance of influencing the capability of the existing facility. The difficulty with this approach is that frequently there are mismatches, leading to money spent for availability where it will not achieve the result because of failures elsewhere. In the Renaissance era, mission systems specified as complete entities offer the opportunity to better address these mission needs through a consistent specification based on operational capability or activity.

For space missions, generally there needs to be high inherent reliability for spacecraft components and short restore times for data operations, along with generally high availability for command and control. Implementations for this need careful consideration. Availability has several notable breakpoints at which costs of implementation substantially changes. These are discussed in the following section under strategy and tools.

H.1.2 Strategy and Tools

There are two basic strategies for achieving a desired availability: reliability and redundancy. The first of these is the most direct: achieve high availability by using components with an inherently high reliability, i.e., lengthen the MTBF. The second strategy is to separate restoration from repair by providing redundant components and the ability to substitute a redundant component for a failed component to restore system capability. In a mission system both strategies are employed, using a variety of implementations. It is trading these to meet a requirement at a minimum cost that is complicated.

In supporting either repair or use of redundant capability, the system must provide effective means to detect the existence of a fault, identify the causes of the fault, and determine the appropriate response to restore system capability. In the Renaissance architecture, this is intended to include system wide knowledge to ensure full information about system activity, and capabilities such as finite state modeling for fault analysis and determination of corrective action.

H.1.2.1 Reliability

All systems require a reasonable level of component and system reliability. If MTBF becomes too short, the system spends too much time in restoration, and too little in its basic operations. Fortunately, existing commercial components have responded to this by providing reasonable reliability.

In some cases, higher than normal reliability is required. This occurs when it is both difficult or costly to repair or replace a failed component, and costly to provide many redundant components for use with no repair capability. The most obvious example of this is the spacecraft. Repair and replacement are at best difficult and costly. Similarly the cost in weight and energy to provide large numbers of redundant components can be very high. For the spacecraft, there is the additional risk that recovery may not be possible for some types of failures because side effects of the failure permanently incapacitate too much of the vehicle. For example, loss of attitude stabilization may result in inability to communicate restoration commands from the ground and in loss of power generation capability.

Use of high reliability components will cost significantly more than standard components. The largest part of this cost arises from the process to achieve the high quality level needed for these. Substantial cost from testing and other verification processes is a given. If unique components are required because of the reliability, the basic fabrication cost will be significantly higher.

H.1.2.2 Redundancy

The most common approach to higher availability is through redundancy. Some missions address this directly by requiring "no single point of failure." This may be overkill when MTBF is years and repair or replacement time is hours. A distributed system allows redundancy as an inherent aspect by providing ready interfaces for redundant elements. There are a number of tactics taken to providing the essential redundancy and failover to such. These are discussed in order of decreasing MTTR, and, partially, by increasing cost.

1. System level cold (warm) backup

   With this tactic, the entire system has a hardware duplicate that is not operational routinely. When a failure in the operational system occurs, the backup is initialized and brought to operational status. Cold backup means there is no power applied to the backup system, while warm means the components have been turned on but not operationally configured. MTTR can range from hours to days, particularly if the backup is remote from the primary system. This approach is usually used only for catastrophic event protection, i.e., a recovery capability for earthquakes, flood and fire. The cost is that of duplicating the system.

   Usually, this tactic is applied at the level of major subsystems to reduce the effort for a transition to backup. However, the MTTR will remain hours to days, and system duplication costs are essentially the same.

2. Replacement level redundancy

   In this tactic redundancy resides with a store of "Logical Replaceable Units" that are not integrated at run time, but are used to replace failed units when failure occurs.

   In hardware, this may mean simply calling a vendor for a replacement (given a service contract with appropriate terms), with the expectation that component MTTR of roughly an hour is acceptable. It can be extended to a local store to reduce MTTR to minutes but requires logistical maintenance of the local store. If replacement forces system re-initialization, MTTR may extend to roughly an hour, even with a local store.

   In software, this will normally mean replacing a file with an image taken from the archive. This may take minutes up to an hour. Alternatively, it may simply mean replacing the executable memory image by re-initializing the associated process. This may take similar times, depending on the complexity of re-initialization, for example, rebooting the computer operating system.

   This approach may actually be less expensive than system level backup, with lower MTTR. The risk is that of misidentification of the cause of failure.

3. Dynamic (Hot) backup

   This usually means that a duplicate exists, already configured and with operational capability but not operationally used. When used at a system level, this is costly because of the full duplication of components and operations. For true hot backup, the duplicate must have access to current system data, usually requiring that it receive and process a significant fraction of the transactions in the operational system. While this allows for very rapid restoration from hardware failure, it raises the risk that both systems fail from an operational problem driven by the incoming data.

4. Shared-Component redundancy

   This tactic is designed around the existence in the active system configuration of more capability than is needed to meet requirements. Then, when a component fails, the

operational activities are switched to that surplus capacity for restored operations capability. Normally, the redundant capability is established at lower levels, e.g., application or server component level so that operational transition is not too complex. This approach is becoming the most common supported commercially.

Manual activity for fault detection and isolation can be used with minimal cost in development, but with higher operational costs and usually having a higher MTTR. So called "high availability" computer systems, from vendors such as Tandem and Stratus, provide for a high level of automated failure detection, isolation and recovery. Such vendors sometimes also include "hot replacement" designs, allowing replacement of failed hardware components, e.g., disks and CPUs, without halting system activity. Similar capability is provided in other unique component types, e.g., network routers and the FDDI LAN technology. With appropriate tools, automated fault detection, isolation and recovery can be extended system wide, providing for very small MTTRs for most failures.

A variant on the redundant capability strategy is used in some RAID types. The parallel disks are used to store redundant data bits, allowing for continued operation in the event of failure for any single disk in the array. These units usually allow hot replacement of a failed disk, and automated recreation of the missing data, without halting the activity within the array.

H.1.2.3 Tools

One of the most important tools is a system management application that supports monitoring of all system performance. This application provides the key interface for collecting information about the status of the system and for issuing corrective actions. The difficulty of isolating problems requires system-level data collection and analysis. Frequently, failure of a single component will affect many activities, making the source hard to identify. Also, operational failures driven by dynamic conditions, like transient data storage and stack limitations, can be hard to identify without full information. The architecture use of SNMP is designed to support system level tools, for example, Hewlett-Packards' OpenView.

The second most important tool is one to analyze the system status to detect and isolate faults and to determine appropriate corrective actions. Examples in use include the Altair MCS finite state modeling tool and Case-based reasoning tools. While simple rule-based tools provide some capability, they are likely to be unable to manage system-wide analysis because of the very large number of rules required.

Other tools available include component-unique tools that will provide for more localized detection and restoration. These include the tools built into the operating system and hardware for Tandem and Stratus, SMP operating systems, database transaction controls and others that are valuable in efficiently managing specific components.

A key to a successful design for a Renaissance system will be configuring the overall system manager to recognize other local managers and to be able to act cooperatively with them. In the near future the limitations of current SNMP implementations will make this especially important.

Extending system management to include ground station equipment and the spacecraft in a single manager will also require integration of the selected application with non-SNMP status and control exchange for these elements.

H.1.3 Alternative Approaches

To illustrate the approaches to meeting availability requirements, the two mission systems shown in Figures 7-1 and 7-2 are used as a basis. Further assumptions about the requirements are made as follows:

- NASA-integrated mission:

  Nominal availability 0.95 during spacecraft communications, 1 hour per day, 7 days per week. Ground system availability 0.95 during other periods, 8 hours per day. Spacecraft availability separately specified at 0.995 over a 1-year period to match requirements for autonomous spacecraft operations.

  MTBF for ground system greater than 24 operational hours, with probability of failure during any 1 hour period less than 0.05. (Note: this implies a MTTR of about 1 hour during spacecraft communications periods.)

  System failure means failure to support spacecraft communications for at least 99.7 percent of the contact period. For non-contact operations, failure is any fault that causes cessation and restart of any ongoing process.

- Autonomous Mission

  Operations to be seven days by 24 hours

  Availability to be at least 0.99 over 3 months, exclusive of weather impacts. MTTR to be less than 5 minutes.

  Maximum of four 1-hour contacts per day

  Availability to customer analysis centers to be under the same parameters as above.

H.1.3.1 NASA-Integrated Mission Example

One key to understanding the requirements for this mission is that the combination of availability and MTBF specifications implies that loss of one contact in twenty is an acceptable operational mode. Depending on the network being used for communications, i.e., SN, DSN or Wallops / Poker Flats, there are some losses for simple weather phenomena that may be a significant contributor to losses. SN has the least impact form this. For SN, there are potential impacts from Shuttle flights that could be significant, depending on the mission needs for SA access. For illustration, assume that the combination of the above causes the loss of one contact in 50.

Combining the availability of space-ground communications with that of the spacecraft leads to the conclusion that the probability of failure of the ground data system must be kept to less than about 0.02 per hour of spacecraft contacts. With availability at 0.95, MTTR remains about an hour for contacts, which basically allows restoration to occur outside of the operational period.

The basic hardware will generally achieve this without too much difficulty since most components post availability at more like 0.9999. The most vulnerable aspects are likely to be the software and operational aspects. Given the allowance of interruptions of roughly seconds (definition of system failure), the best approach is likely to be redundancy of the key software and data elements, with automated switching between these elements for failure of operation. Note that only those components required for space contact operations are included.

If the estimated failure rate is low enough, even this can be avoided. Operational controls may be needed to achieve this, e.g., cleaning up disks and reinitializing operating systems to ensure a clean environment before each contact. If the software and operational failure rates are expected to exceed the needed level, a choice of relatively expensive automated switching tools may be needed. The cost can be minimized by identifying the leading causes and providing automated support around the related components. For example, if the essential problem is that the communications link between ground station and operations center fails, then using a communication server with the ability to automatically re-establish the link would resolve the problem. ISIS is an example of such a server.

Since the availability requirement for other times is approximately the same but the spacecraft and space-ground communications elements are not included, the requirements for the ground data system are relaxed during these periods. Thus, it would be feasible to reconfigure the network to allow some of the redundant components to be used for activities such as maintenance and tests during such periods. Further, the existence of extended periods with no operations (about 15 hours per day) would support both maintenance and testing. Thus, failed components could be replaced during such periods with little need for special effort.

H.1.3.2 Autonomous Mission Example

The requirements above can be met by the design illustrated in Figure H-4. In terms of the hardware architecture, there are two ground stations and the operations and production components all connected by a dual FDDI ring. All components are dual home connected for redundancy. There are three multiprocessor server platforms sharing RAID arrays for most processing and four workstations to support local analysts. Connection to remote analysis sites (and the Launch Facility) is provided through two gateways connected to the FDDI ring. Customer access is provided through the Internet,so that the gateways provide security as well as data routing.

LAN server software resides on each of the three server platforms, including a message server, an ODBMS server and a DFS server. DCE is active on the LAN, with one RAID array reserved for DCE server data, including security and DNS. DNS is maintained with server group names so that applications attach to the group. Network management is performed with an SNMP-based application, using a state manager for analysis and directives. Mission management is performed using an application running on a workstation, using datasets on the RAID arrays. Customer access is provided through WWW servers with access to the product and planning databases.

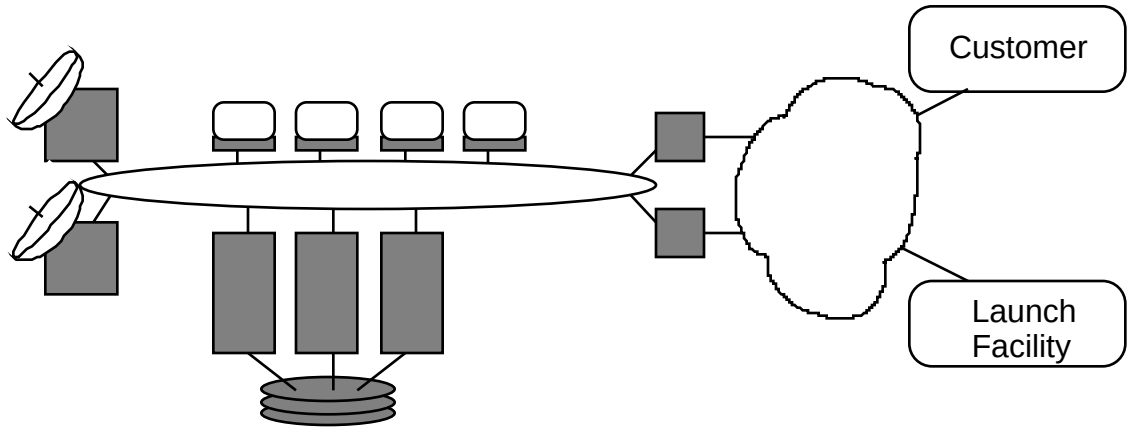


Figure H-4. Autonomous Mission Example Hardware Diagram

This approach combines redundant hardware with local context switching, supplemented by DCE server group technology to provide the effective availability needed. Today, applications fully using DCE capabilities are scarce though anticipated. To avoid writing an interface, alternative approaches can be used, e.g., distributed server applications such as ISIS for message services and a DBMS with distributed server capability. With selected redundancy of data across the RAIDS, NFS servers could be configured to ensure access to essential data.

Maintenance is allowed for by supporting full operational capability on less than the full hardware suite. A server platform and a workstation can be taken off-line for maintenance, and a disk can be extracted from a RAID array or even a full array taken off-line. Alternatively, fully reliant platforms from Tandem or Stratus could be used that have no single point of failure. Similarly, reliant transaction processors could be used as data servers to ensure the high availability, and ensure recovery of on-going activity. These latter approaches would be expected to be more expensive and to be avoided if possible.

H.2 Mission System Performance

System performance addresses the issue of the capability of the system to perform all the activities requested of it in an efficient, effective manner. In this section, we will focus on the ability of the entire set of automated data system components to perform their part. It is recognized that other key elements of performance exist, e.g., the RF signal components, but these lie at a more detailed consideration than addressed here.

H.2.1 Requirements Summary

Data system performance is concerned with the assurance that the system processes data in a timely fashion. The goal is to ensure that the system can act upon data received from the spacecraft or planning sources in a timely enough fashion that appropriate action can be taken. The definition of timeliness varies depending on the nature of the data being processed. The

overall set of requirements from a system viewpoint consists of system response time, system capacity, and system throughput. Each is addressed in a section below.

H.2.1.1 Response Time

The system needs to perform its operations and deliver the required data and control within required time limits. These vary from milliseconds for real-time control, to hours and days for some product generation activities. Response time requirements are attached to specific system-level transactions, defined as initiation to completion activities, and linked to activation frequencies for the transactions. Response time requirements should normally be generated as an expected statistical distribution but can be fixed limits for some (real-time) transactions.

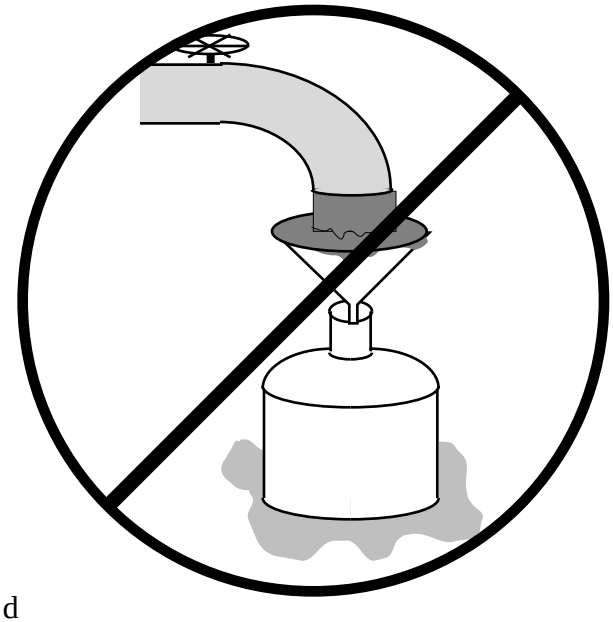
H.2.1.2 Capacity

The system should provide sufficient resources to prevent the system from being swamped and data lost, schematically shown in Figure H-5. The capacity requirements can be operational, i.e., dependent on peak throughput levels, or they can be inherent, as for permanent archive of data. Capacity applies to both passive and active elements, e.g., data storage and processor capacity. Capacity requirements are linked to transactions by the resources consumed to accomplish a system level transaction and the number of activations.

Figure H-5. Eliminate Capacity Overflow

H.2.1.3 Throughput

The system should provide sufficient resources to assure that data are processed and moved through the system in a timely manner. Failure to meet throughput requirements could result in the loss of data, illustrated in Figure H-6. Throughput is related to the transaction activation frequency and the volume of data and processing required per activation.



d

Figure H-6. Eliminate Throughput Overflow

H.2.2 Performance Issues in a Renaissance Environment

In the Renaissance environment, there are several differences from the traditional approach to mission systems. One major change is that the system needs to be assessed as an integrated network of performing elements. The entire network is needed to accomplish the mission objectives. Another difference is that mission systems will have varying control over some parts of the network, i.e., some missions will control the entire network and some will control only limited portions.

Additionally, the use of commercial elements will change the details of what is done to achieve performance. The introduction of multilayered, shared-bandwidth networks will require matching of requirements to existing capabilities that cannot be changed. Conversely, in some cases they may be more changeable, e.g., changing the contract for service with a provider may be easier then rebuilding a network segment.

H.2.3 Strategy and Tools

There are two primary issues that need to be addressed in developing a system with performance in mind: resource contention and workload balancing.

Contention for scarce resources is the primary source of all performance problems. Resources can include the network elements, e.g., routers, gateways, media and interfaces; applications, software services and datasets; platforms, and individual platform components such as disk drives, input/output channels, and memory. In a network of independently initiated activities such as a spacecraft operational network, avoiding contention can be a complex problem.

Balancing workloads across the system elements and across time to meet throughput and response time goals is essential to a cost-effective approach to meeting requirements. It is important to remember that this system is a network and will not perform any better than its weakest element. Enough resources must be applied to meet peak load needs without creating contention, and at the same time, resources should not be wasted through disuse.

There are six general approaches that can be taken to meeting performance goals within the Renaissance architecture. The first two of these are aspects of the specific architecture: selecting the mission hardware to meet mission performance needs and separation the mission system from the requirements of other missions. The other four apply generally to a modern distributed system: segmentation, virtual networks, prioritization and dynamic load balancing.

H.2.3.1 Hardware Performance Selection

The Renaissance architecture has been developed to retain one of the most desirable characteristics of a custom developed system: selection of hardware to meet specific mission performance requirements. This has driven the requirement for "Open System" specifications within the architecture and standards. To achieve this requires a change in development approach in that the software is selected first and then followed by the hardware. This allows the mission to select hardware components with the capacity to meet the capacity and throughput requirements. This includes selecting platform processor and memory capacity, data storage volume and throughput performance for storage devices, and network protocols and media to provide the communication bandwidth and flexibility needed.

Because Renaissance mission systems can be prototyped before mission hardware is purchased, more accurate assessments of problems can be performed and less risk attached to hardware procurement. Prototyping also allows more detailed maps of load to specific components so that generic capacities do not have to be used.

H.2.3.2 Mission Independence

One focus of Renaissance was to make it cost-effective for missions to develop their own system capability. This allows higher flexibility in meeting unique mission needs. It also means that hardware and software component numbers do not have to be oversized because of uncertainties in future, poorly defined missions.

This strategy links with the hardware performance mapping to support a highly effective approach to meeting performance needs without spending money where not needed.

H.2.3.3 Segmentation

Segmentation is the isolation of networks segments to ensure that local traffic stays local. Bridges, routers, and gateways are common approaches used in segmenting networks. The goal of a segmented network is to place platforms that must communicate on a constant basis on the same segment so that two very busy segments will not affect each other. The key to success is to be able to identify data that flows only between a limited cluster of processes. This allows creation of a network in which all data need not be available to all nodes on the network, lowering general performance needs. When the need arises for the segments to share data, a bridge, gateway or router is used to move the shared traffic between the segments.

In the Renaissance architecture, each of the cells is a segment connected by a WAN. The WAN itself may be broken into two (or more) segments. One segment deals with the communication between the ground system and the space system and the other segment deals with the communication between ground-based cells. Within a cell, the network may be further segmented.

Segmentation may be flat, with bridges connecting several backbones. Each backbone would then provide for a single function. Segmentation may also be hierarchical with each backbone providing a higher level of communication. An example of hierarchical segmentation would be to use FDDI to connect file servers and then have application workstations reside on Ethernet segments with a bridge between the FDDI and Ethernet.

H.2.3.4 Virtual Networks

Virtual networks are a relatively recent phenomenon. The virtual network is especially useful in situations where the traffic patterns change with time. Virtual networks use switching technology to interconnect systems. When a connection is needed between two platforms, the switch provides a virtual path between the machines. In the switched environment, platforms can be isolated on a single line to the switch allowing data to travel between nodes at cable rates. Therefore, two platforms connected to a standard Ethernet (e.g., 10BaseT) can communicate at nearly 10 Mbps without interference from traffic from other conversations. The major difference between the virtual network and segmentation is the segments can and do change with time without requiring that the network be re-wired.

H.2.3.5 Prioritization

In some cases, it is not possible to isolate traffic (for example over the WAN). In these cases, traffic congestion becomes a real possibility. Two different approaches to prioritization can be used to mitigate the impact of this congestion. Message prioritization schemes assign a priority to each message that moves through the network. As in computer job scheduling, higher priority messages are given access to the network resources ahead of lower priority messages. Of course, the approach must also give some assurances that lower priority messages are not completely

forgotten. Activity prioritization is more of a scheduling approach. In this scheme, certain activities that can interfere with each other (e.g., receipt of real-time data and transfer of large datasets) need to be managed so that the lower priority activity is not performed or is rate-buffered during the time when the higher priority activity is being performed.

H.2.3.6 Dynamic Load Balancing

Modern computing systems provide functions to dynamically monitor the resource usage across the network and to change allocation of software activity to resources with less load. A common form of this is performed at the platform level of a multiprocessor platform. The multiprocessor can be run either as independent processors, with high-level allocation to individual processors, or, more efficiently, as a Symmetric Multi-Processor, with low-level load balancing between processors. Similarly, network management systems support node level reallocation of processing. For Renaissance, by separating the applications from the operator interface, application activity can be moved from node to node without impacting the operator. Additionally, static datasets and software images can be replicated to meet changing demand. For fully automated activity, such reallocation is easier. These re-allocations generally also take into account the network loading, and can be quite complex decision making systems.

H.2.4 Approach Alternatives

The most important aspect of system performance is understanding the performance limitations of the system and providing methods to work around those limitations. In any approach, the strategies outlined in the previous section can be used within the cell. The biggest differentiator between approaches is in the ways that WAN performance is addressed.

H.2.4.1 NASA-Integrated Mission Approach

In this system alternative, shown in Figure H-1, the system performance is bounded by the performance of the resources that are provided by NASA. The WAN and the GSFC MODNET, on both the protected and public sides, are shared by a number of other projects. The mission system must be protected from performance impacts caused by traffic from outside the mission system as well as ensure that it does not impact other mission systems performance.

Installing protocol and address filters at the network connect points will ensure that data intended to stay on the local area network remains there. They will also ensure that data not intended for the LAN will not enter the LAN. A router serves nicely as such a filter.

The protected network is a special cause of concern since many missions will be using this network to move data from the spacecraft to the ground system cells. Since the protocols to be used on this network allow shared bandwidth, it is important for the mission system to be aware of the impact that other people’s traffic has on their own performance. The mission will need to negotiate with the MODNET management to ensure that it receives the bandwidth that is needed to carry out its mission. In addition, the mission system management will need to monitor traffic that its local cells put on the WAN to ensure that only necessary traffic is being shared between cells.

Particular attention should be made to reducing the amount of time-critical data that travels across the WAN. Placing resources at appropriate places in the system can eliminate much of this traffic. For example, a mission may need to carry out testing when the spacecraft is at the launch facility. Rather than conducting these tests across the network, a workstation could be placed at the launch site. This workstation could then automatically conduct the tests and the results could be sent back to the control center via bulk data transfer (e.g., ftp or electronic mail).

The ability to gain from virtual networks is limited because of the use of the shared network. Virtual networks can be used within a cell, but this may not help as much as would otherwise be possible. Similarly prioritization of data transport is dependent on the implementation of the shared network and its ability to use priorities.

H.2.4.2 Autonomous Mission Approach

In this approach, shown in Figure H-2, the mission system is solely responsible for performance. The emphasis in this approach should be on providing a network that meets the performance goals at the optimal cost. In this example, maximum use can be made of the hardware selection options built into the Renaissance architecture.

Segmentation has less of a role in this system, which because of scale and perhaps location can be physically more co-located. There is still a role in separating the time critical from that which is not time critical, to minimize needed network resources. Conversely, dynamic load balancing can play a larger role since the mission has more control over the used resources and because they are more co-located. Virtual networks can be effectively used to minimize the base performance rating of the communications network and can effectively work with load balancing to optimize use of resources over changing operations modes.

Since a mission system can never be completely isolated from shared networks, some of the same schemes used in the NASA-integrated approach are still valid. Since the shared network most likely will be the public Internet, guaranteed bandwidth may not be available, and guaranteed response time is impossible because of the dynamic routing. As little as possible time-critical data should be sent over the Internet.

As shown in Figure H-2, the mission system may also contain some dedicated WAN connections. These are most likely to be provided by a third party with a recurring cost associated. A major reason for providing these dedicated links is to ensure that adequate bandwidth is available to the mission for time-critical functions at optimal cost. Several options can be used to meet this goal. Nonreal-time data can be stored at the ground station and transferred at times when the link is quiet. In this case, the link only needs to be able to support the real-time rates. The links do not have to be dedicated circuits in the future. Virtual circuits can be arranged (over ATM) with the cost based on the amount of use. The public Internet can be used to transfer non-time-critical data while the dedicated links can be devoted to the transfer of time-critical data.

H.3 Architecture Approaches to Security

This section describes the approaches to mission system security. Sections sequentially describe the requirements, the strategy and tools for meeting mission requirements, and finally, the specific approaches mapped to the illustrative mission systems.

H.3.1 Requirements Summary

H.3.1.1 Data Privacy

The system must provide appropriate measures to prevent unauthorized reading and copying of data identified as private. Private data can include customer data, either payload results, derivatives of payload data, or other data accessible in the general system. Private data may also include details of system addressing mechanisms, passwords, command structures and other data items that are deemed by the mission to be kept from public access. Finally, private data may include other data accessible over the network that is not directly mission related.

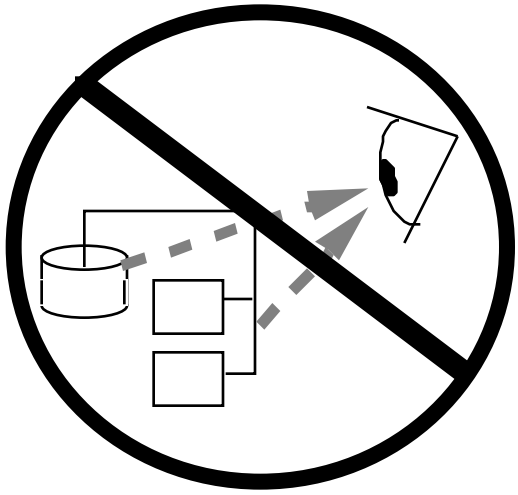


Figure H-7. Security Provides Data Privacy

H.3.1.2 Resource Protection

The system must provide measures to prevent the accidental or deliberate destruction of system resources by unauthorized persons. System resources include physical resources, e.g., the spacecraft and other computer controlled mission resources, computer managed data and information both internal to the mission and outside, i.e., on other NASA systems such as the Space Network and NCC, and customer systems.

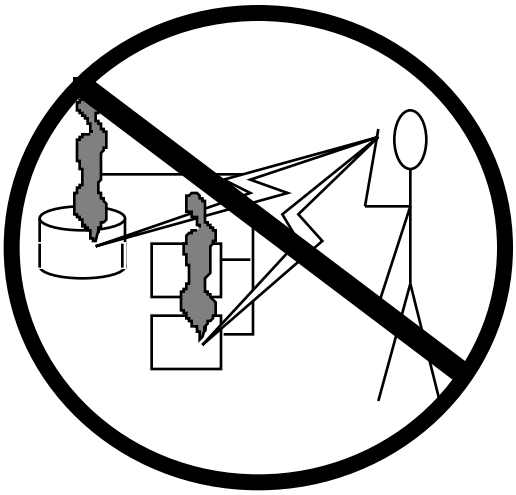


Figure H-8. Security Provides Resource Protection

H.3.1.3 Service Assurance

The system must provide measures to ensure that the capabilities of the system are available for mission use as planned, i.e., not subverted by accidental or deliberate misuse of resources. This includes ensuring that communications, computer resources, data and personnel support are available as planned for mission use. This obviously overlaps resource protection in that destroyed or damaged resources will not be available.

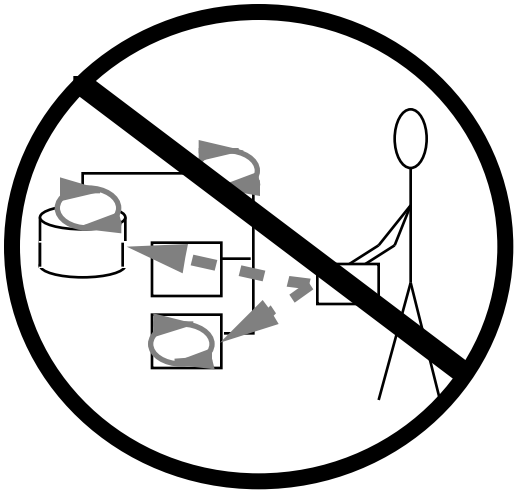


Figure H-9. Security Assures Service Availability

H.3.2 Threats in a Renaissance Environment

In the Renaissance environment, two conditions common to NASA-run missions will change, introducing new sources of threats against the mission system. These are the integration of the

mission system into public networks, including the Internet, and the operation of the mission in locations that may not have the physical isolation and control associated with operations within a NASA site such as GSFC.

H.3.2.1 External Threats

Both these factors raise the level of threat resulting from accidental or deliberate actions by persons external to the mission, and the operating environment. Electronic integration with the Internet is done to provide easy access by some customers. However, it also means that the tools exist for outside persons to have access to, and use, system resources, and that many people are using some elements of the integrated network regularly. Such outsiders are a threat for all three of the security concerns: data privacy, resource protection and service assurance.

External access to system resources can be direct, using Internet tools to acquire legitimate account identifiers and passwords, e.g., anonymous FTP, or backdoor through illegitimate intrusion to acquire access by various means. The security requirements can be overcome through such means as address spoofing, packet replication, illegitimate acquisition of account identifiers and passwords, and through injection of agents such as viruses and worms.

External access can also come because of the lack of physical isolation. If there is public access to the operations facilities, more means become available through logging onto mission workstations, physical intrusions such as wiretaps, and simple physical destruction. Quite aside from deliberate destruction, one of the biggest hazards for any workstation is a cup of coffee in the hands of a tired worker.

H.3.2.2 Internal Threats

Aside from outside threats, security violations can arise from sources inside the mission system and team. In many ways protection from internal sources is more difficult because they will have more opportunity for contact with system resources. Persons on the team can accidentally, or deliberately, cause problems with the system. This can lead to unique opportunities for problems through error, through deliberate injection of time bombs, trapdoor programs and Trojan horses, and through deliberate destruction and creation of illegitimate data copies. This means that simple isolation from external elements is not sufficient, and there must be some means of limiting the violations internal to the system.

H.3.3 Strategy and Tools

There are three basic approaches used to embed security in a computer network system. These are access control, activity audit analysis, and data encryption. Access control means simply the implementation of rules and constraints on access to system resources. Activity audit analysis is a process to identify attempts at security violations, with the intent that access control mechanisms can be manipulated to thwart identified attempts. Data encryption is a means of ensuring data privacy, both for its own sake and to prevent the misuse of information such as passwords or other access authentication information. The implementation of these results in a

stack of security processes, as illustrated in Figure H-10. Each element in the stack can be implemented with varying levels of trust to meet the specific needs of the mission.

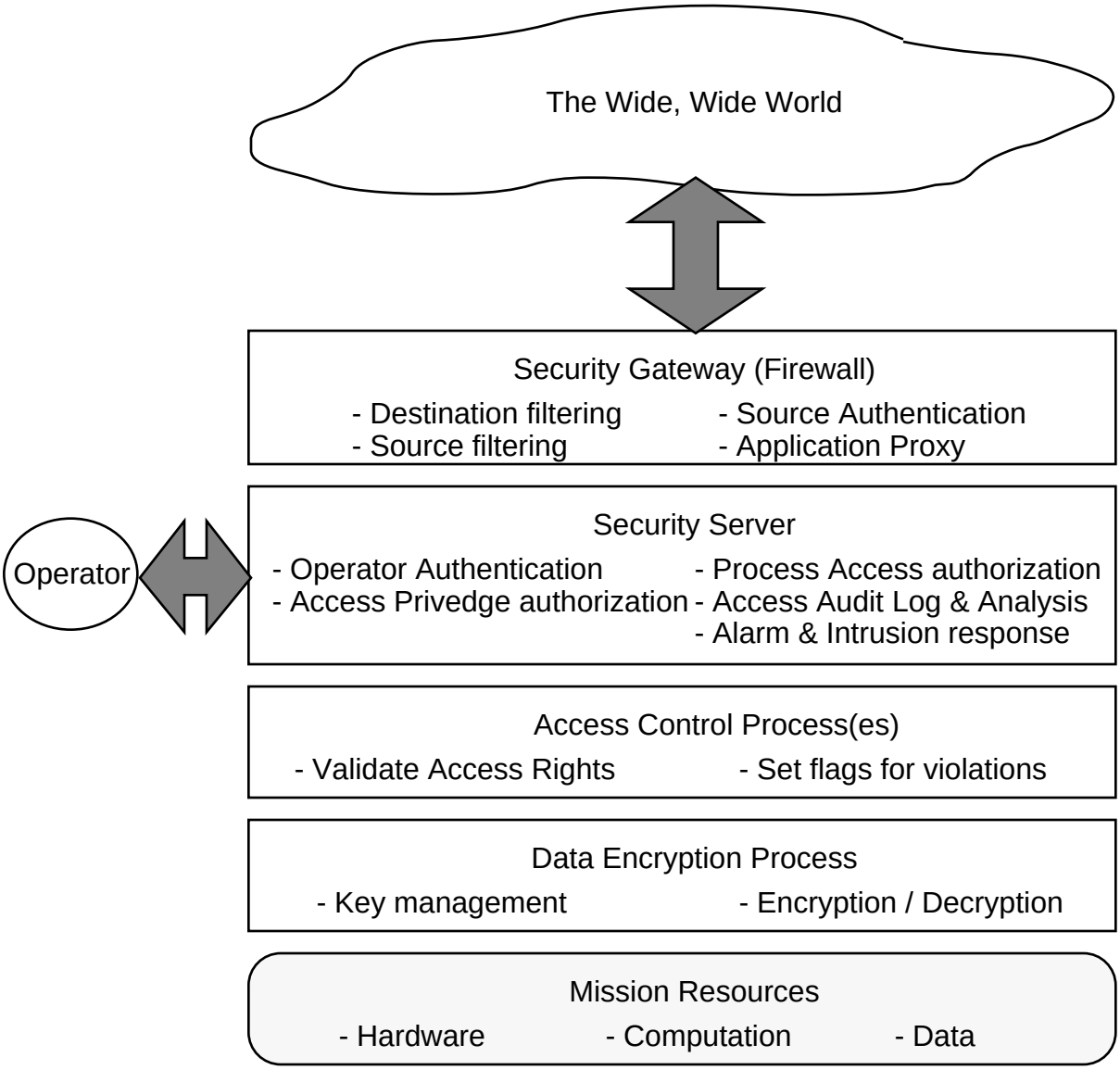


Figure H-10. Security Process Layered Model

H.3.3.1 Access Control

The implementation of access control is probably the arena of most intense activity because it has the most direct impact in preventing ill results from attempted security violations. It provides a means of preventing illegitimate access by outsiders, and of limiting the problem of internal violations. Access control can be broken into three basic components: access authority authentication, access authority enforcement, and intrusion detection and termination.

Access authority authentication is the process of ensuring that the person and/or process trying to access system resources is accurately identified, and has the access privileges for what is being attempted. In a distributed system this becomes very complex and difficult, especially when remote access is present.

The simplest form of access control is a filter that acts to reject all access attempts not clearly identified with a legitimate source. This is the simplest form of security gateway, commonly referred to as a firewall. The next level in control is authentication of the identified source of access.

If a remote operator is attempting to log into the system, the system must ensure that the logon is, in fact, from the identified person. Passwords are the traditional means of authenticating the operator, with physical adjuncts, e.g., smart cards or others, used to supplement the mechanism and avoid illegitimate use of stolen passwords. Similarly, encrypted passwords are used, based on the assumption that both the encryption algorithm and key must be known to provide this. In situations with data privacy, the system must further authenticate the location of the access, i.e., that the operator is located on a legitimate system resource for the desired access privileges. This too is frequently done through the exchange of "passwords" between system hosts to deter address spoofing. Finally, the system must authenticate that the process performing the specific activity has the access linkage to an authenticated operator by maintaining an authenticated link between the process and the operator on whose behalf is acting. The use of client-server processes, in which an in-system client connects to an external server to read or write data, makes this more complex.

Process authentication has been implemented commercially in several ways. Some Security Gateways implement session and application specific authentication. The Distributed Computing Environment provides authentication by mapping through the Kerberos security server to get time dependent access keys for processes acting within the system. More complex approaches have been taken in developing operating systems that support multi-level security. These have their own internal security processing with a more complex mapping of access privileges to persons, processes and data, and some versions are commercially available. Generally, the security process and databases are isolated from the rest of the system to prevent subversion, and in some cases, changing the security process or data may require a hardware change for the same reason. It should be noted that these dense security approaches require extensive computer and human resources to perform, and can impose a significant burden on the system operators.

H.3.3.2 Activity Audit Analysis

Analysis of system activities is performed based on selected audit data that is collected in real-time. The analysis may be performed either in real-time, or off-line from the audit log. The former is performed by systems with the intent of detecting intrusive attacks while they are happening, and being able to change system operating conditions so as to thwart detected violation. Off-line analysis is performed to examine the sequence of activities which may indicate the early stages of an attempted break-in, and to look for activities which may indicate the need for changes in security policy and related actions.

The collection of audit data is key to much of the long term success of the security approach because of the ever changing nature of the techniques used to break into systems, and the variations in activity that can cause problems based on unique characteristics of the system. From the security manager's point of view, the more data collected the better. However, it has been amply demonstrated that collection of audit data can effectively be used to consume all the resources available on the system, thereby achieving the very results it attempts to prevent - service denial. For every system there must be an analysis of the threats expected to be most severe, and the minimum audit data required to detect those threats. There may also be periods where planned usage is low, and additional data can be collected to try to detect unexpected events. The balance between security and operational effectiveness is continually changing, and this has to be accepted.

H.3.3.3 Data Encryption

Data encryption is an approach used to protect data for a variety of purposes, and can be further used as a source authentication approach because of the need to use appropriate encryption / decryption keys for valid data interpretation. Digital signatures are an example of the latter purpose. This use indirectly acts as a proof against error or data tampering, since changed bits will be detectable after decryption.

Various encryption schemes and standards exist. The public mechanisms are generally useful in providing protection against casual access, but are breakable with skilled intent. The PGP is a better algorithm than DES, but still breakable, by intent of the US government. Some usage within some government agencies requires use of better algorithms, which will normally be provided by the US government, and operate under special constraints.

One of the less apparent difficulties of encryption use is encryption key management. Both sender and receiver must agree on the key in use for each source and destination, and time of transmission. If a system has many paths in use at a time, there can be a complex activity devoted to ensuring that all key usage is coordinated and accurate. Additionally, key distribution must be protected to avoid use by unauthorized persons. Currently, use of public key algorithms and smart cards has simplified much of this.

H.3.4 Approach Alternatives

The first, and probably most important, element of a security approach is to devise and document a consistent, system wide strategy. This security plan should identify the risks and level of protection desired for the mission, and define the approach to be taken for all the distributed facilities associated with the mission so that a consistent level of protection is provided. Any security approach is no stronger than its weakest link, and holes at the system level will drive added cost to repair these, with a less satisfactory result in both security and operational performance.

With different missions, different levels of security will be appropriate. NASA prescribes methods according to four Sensitivity levels, 0 - 3, with 3 requiring the most stringent methods (NASA Automated Information Security Handbook NHB 2410.9). Missions using NASA resources, and sponsored by NASA must respond to this identification. Most mission systems

will correspond to levels 1 or 2, but major NASA systems are at level 3. As a result this architecture provides multiple levels of applicable security to match, with several variations in architecture for the needed approach. Additionally, when the differing mission system designs are reflected, the security approaches must change to adapt to these.

To illustrate the alternatives, two different mission system profiles are used as a basis for exploration of requirements and the alternative approaches to meeting them. The first alternative is the "traditional" GSFC mission profile: one using NASA resources for space to ground, and for terrestrial communications; and using other NASA resources for launch and flight operations. The second alternative is placed at the other extreme: a self-contained, autonomous mission system that uses no NASA resources in flight, and only links for launch support, e.g., Conestoga at Wallops.

H.3.4.1 NASA-Integrated Mission Approach

The example mission system, as shown in Figure H-1, poses two primary security issues for the mission developer. First is the integration of mission security in the larger context of NASA provided support systems, and, second, the integration of customers with network connectivity with varying levels of security external to the mission unique aspects.

The NASA context provides standard, existing security support in many areas. Physical security is provided by site access controls for the overall facility, and badge control provides mechanisms for more restrictive building access to an operational area. NASA provides a secured network, and secured facilities for launch, communications, and some NASA-internal customers. In the GSFC environment, there exists the MODNET, with controlled connectivity and a security gateway for connectivity with others outside the controlled environment. Finally, many external customers connect to the GSFC network through a leased line arrangement, which provides a high level of access control.

For a mission operations area at GSFC, connected to the secured side of MODNET, the threat of outside access to the system resources is minimized by the MODNET security gateway, and the controlled access on the secured side. Generally, only a limited access control process is needed within the mission operations system, to identify operators and control access to data and resources. If the mission is categorized as NASA Sensitivity Level 3, then more restrictions are required, and a combination of physical access control, a security gateway and internal access control mechanisms such as DCE are needed for the operations area. NASA resources to date do not use encrypted data for their interfaces to missions, and are not planning to do so in the future.

The interface of the mission operations system to customers and mission operations activities, e.g., science planning and/or analysis, which are located elsewhere is more complex. The primary risks are read access to transmitted data, injection of erroneous data, and injection of data that is harmless but consumes many system resources.

The Security Gateway on MODNET provides an effective filter to eliminate intrusion of invalid messages that consume mission system resources. Thus, the mission solutions start with an appropriate mechanism for operator authentication for the remote operators. For most uses, the SmartCard mechanism is acceptable for direct access, and is currently included in the MODNET

gateway implementation. For indirect access, i.e., process to process exchange, the remote system must have a means of authenticating its privileges, either by digital signature or other encryption techniques. Use of full encryption provides a mechanism to ensure that data is received and accepted only if unmodified from the authorized source. Encryption also ensures that private data is not read by unauthorized persons.

For a specific outside access, the choice among these is based on expected threat, and services needed. The intent is to minimize security overhead by addressing only real threats. Note that all of these mechanisms presume an appropriate level of security at the remote site itself. In a hierarchy of increasing trust, the following measures can be invoked:

- MODNET filter plus standard OS access control, e.g., ftp only with operator ID and password. (Discretionary Access Control)
- Operator ID using SmartCard authentication (Level 2 systems)
- Use of digital signatures for data sets transferred.
- Commercial encryption of data sets
- Additional security gateway with application proxies
- DCE/KERBEROS access control
- MLS (Mandatory Access Control) and high level encryption (Level 3 systems only)

The Mission Operations area security process stack will essentially include all but the security gateway layer, which is provided by the MODNET gateway. Additionally, the stringency level of these applications will be reduced generally by the fact of being within the MODNET environment, which provides supplemental security processing.

H.3.4.2 Autonomous Mission System

For the autonomous mission system illustrated in Figure H-2, the provision of security lies entirely upon the mission. While this implies a more extended set of security tools, the reduced number of interfaces with differing trust drives towards what is in many ways a simpler security system.

The single largest difference is that the mission must provide for full physical security as well as computer security. In a campus environment this can be difficult. Both the actual operating area and the communication lines must be protected at an acceptable level. Physical access provides the most direct routes to both resource destruction and bypassing of security measures.

Because of the direct link to the Internet, the minimum security is different for this system type. The Mission Operations facility should be separated from the Internet by a high quality security gateway, with application proxies for all internet applications that are needed to communicate across the Internet to data customers or the launch facility. Additionally, the minimum facility operator authentication should include a controlled password mechanism for internal operators, and a SmartCard mechanism for remote operators. In general the security layers will contain the complete set shown in Figure H-10.

If the Ground Station is local (as illustrated), it can be directly integrated with the access control mechanisms of the Mission Operations facility, and require no additional security processes. Otherwise it must be armored in the same way as the Mission Operations facility, albeit with a more limited set of operators and applications required to pass the gateways.

The launch facility will need to support the same type of security processes as the Mission Operations facility, though again with a differing set of authorized operators and Internet applications. Depending on the facility, the mission components may be either integrated with other facility systems, or separate. If integrated, then similar issues to those discussed for the NASA-integrated system must be addressed in establishing the mission components.

Customer sites which receive data will need to incorporate gateway and operator authentication processes in the same way as the Mission Operations facility. If the received data is always a replicate, without stringent survival requirements, then these may not be as demanding as for the Mission Operations facility. If archival is required, then the more stringent approach for operator authentication must be used to prevent destruction of data.

Customer sites that also provide planning and control data must have both a local operator authentication process, and the capability to include digital signatures for data sets transmitted to the Mission Operations facility.

A local customer site may have local connections to the Mission Operations facility. However, unless it has the equivalent security gateway and internal security processes, the communications between them must proceed through a gateway on the operations side to protect the operations resources. In other words, a security perimeter can be established to either include all or part of the customer site along with the operations facility. For example, if there is a single customer site for payload analysis and real-time commanding of the payload, it would be best to include it within the operations security perimeter for performance reasons, and to minimize the chance of intrusions impacting operation of the payload.

For systems with more stringent needs, there is a hierarchy of added functions similar to that for NASA-integrated systems:

- Use of digital signatures for data sets transferred.
- Commercial encryption of data sets
- DCE/KERBEROS access control
- Additional security gateway process for full encryption and address hiding (Level 3 systems only)
- MLS (Mandatory Access Control) and high level internal encryption

Encryption of data sets is performed for data privacy, and as an additional protection against subsequent intrusion using information in the legitimate messages. DCE/KERBEROS is used for extended protection against unauthorized intrusion, through the use of time-dependent access tokens. Level 3 Sensitivity systems require extended specialized security measures to raise the level of protection against both data exposure and system intrusion.

H.4 Mission Data Quality

Data quality is a critical measure of the success of a mission system. The goal of the mission system is to capture the data from the customer operations, and to transmit this data to the customer in a permanent form in time to support customer needs. Data quality is an integral metric of the products delivered to the customer, and has indirect impacts on the ability of the mission system to operate so as to support desired customer operations, i.e., there is a potential impact on the ability to perform the necessary system operations.

H.4.1 Requirements Summary

For the customer, data quality applies to both products, and to the operation of the spacecraft and instrument to ensure both the expected environmental conditions, and the proper control of the instrument in performing a data collection operation. For the products, the ultimate desire is for complete, error free data sets. The goal for spacecraft operations is that the environment for the instrument be within the limits required for effective data operations, and that the important environment metrics be captured during a data operation. The goal for data operations is the customer instructions to the instrument are reliably supported, so that operational errors are not introduced. In the rest of this section, we will address only the aspects of these which relate to data transport and capture, and the error problems with this. The other operations aspects are partially addressed in the earlier sections of this Technical Performance section, and other parts are linked to the operations concepts.

Data errors can be divided into three basic categories: lost data, errors that are detected and linked with specific data elements, and undetected data errors. The detected data error category has two subcategories, those which can be corrected, and those which cannot. The characteristics of these are mapped against the potential impacts in Table H-1. As shown, the sensitivity of the mission to types of data quality errors varies depending on the nature of the mission and its operations. As a result, the approach taken by the mission to effectively respond to potential errors will depend on the nature of the mission.

Table H-1 . Data Quality Sensitivity

Quality Problem	Data Redundancy	Operations Directness
Data Losses - Size	Missions with high redundancy within a measurement may tolerate small gaps. Low redundancy measurements will not, a gap may spoil the entire measurement.	The impact on operations depends on the response time needed. During a critical time, gaps in command streams and down-link data may be disastrous.
- Frequency	Larger gaps may be tolerated if measurements are repeatable, and in scope for the mission	An "autonomous" mission is built to not require direct contact. For these, gaps are less serious
Detected Errors - Corrected	For products, corrected errors will make no difference	Corrected errors cause no problems unless a process fails because of error processing.
- Uncorrected	As with gaps, errors can be tolerated if there is redundancy within the measurement.	Uncorrected errors are usually not significant on downlink. Errors in commands can be serious.
Undetected Errors	Undetected errors can be serious in causing erroneous results. If data is redundant, then sensitivity is less, but can cause much effort to identify the source of the data difficulties.	Undetected errors in command images can be very serious. Usually, extreme lengths are taken to avoid this. On downlink, there is usually enough redundancy to avoid serious problems, but it can cause delays in solving a problem.

H.4.2 Strategy and Tools

Table H-2 shows the areas that can affect data quality and approaches that can be used to increase data quality or to recover from data quality problems.

The approach used to correct a problem in data quality needs to be weighed against other factors than just the completeness of the archive. These factors include the cost to recover the lost data, the timeliness of the data, and the sufficiency of the data. In some cases, the costs involved in avoiding or recovering from data loss may not justify the effort. In others, providing some of the data in time for a particular event is more useful (or the only ultimate use) than providing a complete data set at a later time. It may be sufficient to make analyses or proceed with other processing with a significant fraction of the data missing. Finally, the nature of the data needs to be well understood. Some data is less useful if large contiguous pieces are missing, than if one

or two records are missing. Other data has the opposite qualities. All of these factors need to be weighed to determine which approaches should be taken for a specific mission.

Table H-2. Factors Affecting Data Quality

Where	Causes	How Recovered or Avoided
Spacecraft	On-board image hits Space/Ground Link Errors	Image error detection & correction Image Dumps and retransmission
Space/Ground Link	Poor signal quality Loss of Signal	Error Detection - CRC Error Correction - Reed Solomon Higher Level Error Tolerant Protocols over the space link - TCP/IP
WAN	Data loss in transmission	Higher Level Error Tolerant Protocols - TCP/IP
LAN	Data loss in transmission	Link Layer Error Correcting Protocols - Ethernet, FDDI Higher Level Error Tolerant Protocols - TCP/IP
Storage	Media Failure	Redundant Storage - RAID Backup - Tape, Disk Stable Long-term Archive Media - Optical Disk Archive Management
Processing	Improper setup Component failure	Reprocess data (data must have been recorded upstream of the processing Process recovery (See section H.1)

H.4.2.1 Spacecraft

Problems with command data can be resolved in a number of ways. Various data transmission methods (see section on space-ground link problems) can be used to ensure that the data is transmitted to the spacecraft without error (or at least that errors are detected and corrected or the commands in error are resent). The on-board process and data storage can use error detection and correction encoding, e.g., parity bit storage and processing. Also, portions of memory can be dumped back to the ground for comparison with expected values, and data re-transmitted as needed. For collected payload data, data storage error correction can be used on-board, and encoded in the data transmitted for ground error correction.

H.4.2.2 Space-Ground Link

The space-ground link provides the most likely point at which data obtained on the spacecraft will be lost in the mission system. The link is temporary, sometimes difficult to initiate and maintain, and can have a poor signal to noise ratio, or have RF interference problems. As a result, data quality over this link is highly time dependent. Consequently, the link can affect the

overall data quality by creating gaps in the data stream caused by loss of signal, and by injecting errors because of poor signal quality.

A number of approaches are already in place for dealing with corrupted data on the space-ground link. At the very least, most missions use a cyclical redundancy check to determine if the data has been corrupted. Data failing CRC is typically discarded. At the next level, missions may use Reed-Solomon encoding not only to detect errors, but also to correct them. Both of these approaches will detect and eliminate errors in the data, and Reed-Solomon significantly reduces the amount of data with low error rates. These do not address recovering data that is lost in transmission.

Gaps caused by very poor signal quality can be recovered by retransmission from the spacecraft. Today this is performed in bulk, by operator command to the spacecraft. In the future, higher-level error-tolerant protocols such as TCP/IP may be pushed out to the spacecraft to support automatic retransmission at the packet level. The Renaissance architecture supports this concept by pushing these protocols out to the ground stations, thereby enabling the push of the technology to the spacecraft.

Since data storage is a non-extensible resource on the spacecraft, trades must be done in the mission system to determine what to do if data stored has not been transmitted, and no storage remains available. These trades must consider whether or not the spacecraft should continue to collect data (and overwrite already gathered data) as well as what operational steps should be taken to retrieve the data (e.g., using a backup station).

If the mission system is intolerant of data loss, then providing sufficient storage on the spacecraft to allow for missed or lost contacts should be considered. The trade must be made to determine how tolerable losses are versus the additional costs of adding storage to the spacecraft (dollars and weight).

H.4.2.3 WAN

Historically, the wide area network has been a source of concern in data quality. The operations network is currently a network of serial matrix switches connecting unique data paths. No protocol support for error detection and correction is provided, and management of routing on this network is manually intensive. As a result, transmission routinely introduces some level of errors into the data, and gaps can be created by network component failure.

The Renaissance architecture uses Internet protocols for the transmission of data over the wide area network, thus providing a commercial source for inherently high quality components. Additionally, IP products support high availability through automated re-routing in the event of component failure. TCP is recommended to remove transmission errors by retransmitting corrupted or missing packets. However, the use of TCP for high volume, long distance transfers has performance limitations.

The data transmission mechanisms need to take into account that, during a spacecraft contact, real-time data may need to reach the appropriate destinations as quickly as possible to support decision making. However, the system should not sacrifice the completeness of the playback

dataset for the timeliness of the real-time data. Resource-constrained missions may consider the option of storing playback data at the ground station during contact to avoid WAN performance problems, and forward the data at a reduced rate.

H.4.2.4 LAN

Data quality on the local area network is not much of a concern anymore. Most mission systems have adopted commercial networking technologies such as Ethernet and FDDI as well as Internet protocols to eliminate the loss and corruption of data during transmission on the LAN. There is still a potential for losses because of dropped interprocess communications. This is avoided by use of automated recovery techniques.

Timeliness can be a concern on the LAN. LAN performance was addressed in section 7.2.

H.4.2.5 Storage

Data storage is also a potential source of poor data quality through partial or full file losses. Several technologies can easily be fit into the Renaissance architecture to reduce the impact of a media failure on the data quality: RAID, hierarchical server management (beginning to make some headway from the mainframe into the distributed market), traditional backups to tape, and the use of long-term, stable archive media such as optical disk.

H.4.2.6 Processing

Data processing is often overlooked as a source of reduced data quality. However, data processing can affect the data quality in two ways: improperly setup applications and process operational failure. As long as the input data is preserved, these errors can be corrected fairly easily by reprocessing the data with the correct setup. However, timeliness of the output is dependent on system availability tools as discussed in section 7.1.

H.4.3 Alternative Approaches to Data Quality

For purposes of illustration of some alternatives for addressing requirements for data quality, we will analyze the two sample mission systems shown in Figures 7-1 and 7-2. For this purpose, it is assumed that the NASA-integrated and autonomous mission systems have the following data quality requirements:

NASA-Integrated Mission

- Collect a minimum of 95 percent of mission observations, where:
- Each 1-hour observation has less than 10**-10 error rate, and no data gaps
- Operational control requires 10 minutes / day of real-time control with an error rate less than 10**-10 for commands and 10**-8 for monitor data.

Autonomous Mission

- Collect a minimum of 90 percent of 4-hour collection periods, where:
- Each period has an error rate less than 10**-8 for captured data, and no data gap is greater than 1 minute.
- Collection gaps greater than 8 hours are not acceptable
- No real-time control requirements exist after L&EO

These differing requirements point to differences in desired approaches. The impacts on each mission and are analyzed below.

H.4.3.1 NASA-Integrated Mission System Approach

The first impact of the observation error rate is that Reed-Solomon encoding, or equivalent, must be used to achieve this level of observation quality. Second, the observation must be assessed as a unit. This points to being able to manage errors and retransmissions of data at the observation data unit level. If IP protocols are extended to the spacecraft, then packet level replay, linked to Reed-Solomon detected but uncorrected errors, can be used to produce the desired error level.

The Observation collection rate of 95 percent probably means that missed spacecraft contacts need not be operationally rescheduled, but that procedures and resources must be present to allow that in the event of lower than expected system performance. If TDRS is being used, the competition with a Shuttle mission may lead to a need for rescheduling contacts to avoid too much loss.

The ground network is presumed able to use TCP/IP for product data transmission. This will ensure transmission of error free data at that level, and avoid the need for intense operational activity to manually retransmit data on the ground. The one requirement that comes form this is that the ground network must be able to buffer data so that component failure does not lead to data loss, and the operation of these buffers is reliable.

In the data production subsystem, this implies redundant storage for product data, at least until delivered to the customer. Additionally, there must be accounting to ensure that delivered products match with that captured, and that error rates in processing are validated.

The requirement for 10 minutes per day of real-time contact will require the presence in the real-time command and control subsystem of hot backup capability for that desired time period. The limit to 10 minutes per day implies that the components used for this are available for other use for most of the time, which will limit the need for additional hardware and software.

The error requirements for command and control imply a need for closed loop validation of correct receipt, at the spacecraft, of commands sent. In today's world, this can only be done by having the spacecraft transmit received commands to the ground for verification, with sufficient performance to allow retransmission of those found to be erroneously received. In the future, the use of protocols such as TCP/IP, and for more on-board analysis capability may allow for automated retransmission of erroneous commands. This would simplify the ground system requirements considerably, in that the closed loop performance would be less intense.

H.4.3.2 Autonomous Mission Approach

The combination of 90 percent capture and allowance of individual gaps up to 1 minute allows a more relaxed approach to data capture. Retransmission from the spacecraft is unlikely to be needed. While a base error rate of 10**-8 may allow avoidance of Reed-Solomon encoding, this is not a large cost issue, and use of Reed-Solomon can provide a safety factor in both correction and detection of errors. The allowance of data gaps up to 8 hours leads to the freedom from tight processing and scheduling of spacecraft communication. Contacts that are missed need not be rescheduled, unless there is an on-going problem that would lead to major loss of data.

The lack of any real-time requirement for normal operations after L&EO can support a change in mode of the transition between I&T and operations phases. Using the I&T equipment to support L&EO, followed by replacement with a simpler automated analysis, command and data production process would be feasible. Operations would be scheduled at the convenience of the staff, and might be remote. The data capture capability would be the most constrained part of the system. A future system using IP to the spacecraft would support use of processes like FTP for both command uplink and both engineering and payload data downlink. With a beacon-mode of operation for the spacecraft, most of the operation could be fully automated. Only mission planning needs active human interaction on the control side, and analysis on the downlink side. The latter would include analysis of detected anomalies from the engineering side, and production statistics monitoring from data capture.

The need for reliable data capture means that the ground system needs effective, non-volatile data buffers, but allows for a relatively slow recovery from errors given a generally long MTBF. In an autonomous system, this would lead to placing a data store very early in the stream, with highly reliable storage capability, e.g., a multiprocessor system, feeding a RAID storage device. A MTTR of hours could be allowed, with human replacement of failed hardware components, and routine reinitialization for software failures. The other requirement on the system is a highly reliable capability of detecting problems, and notifying the operator backup for both long term problems in operations, or for failed components.

H.5 Summary

As shown in the previous four sections, the approaches to providing the differing forms of technical performance overlap. Distributed servers provide both higher availability and better performance. Use of security features may have an adverse effect on performance because of the addition of processing for filtering and auditing. In general, the issues of security and availability should be addressed first. Then performance and data quality can be addressed in the context created by solving the first two.

One common theme has been followed in creating this Generic Architecture: Implementation must be feasible with existing, off the shelf products. The strategies and approaches for achieving desired technical performance have followed this. While not all combinations are available, and choices do have cost implications, these approaches can be implemented with available products, both hardware and software. Given the current commercial interest in the

issues of technical performance in distributed systems, one can anticipate a more plentiful and capable suite of tools in the future.

Abbreviations and Acronyms

3D	Three Dimensional
ADC	Analog to Digital Converter
AFS	Andrew File System
Altair MCS	Altair Mission Control System
AM-1	Ante Meridian 1 (one of the EOS satellites)
API	Application Program Interface
ASCII	American Standard Code for Information Interchange
ASIST	Advanced Spacecraft Integration and System Test System
ATM	Asynchronous Transfer Mode
BACP	Bandwidth Allocation Control Protocol
BBN	Bolt Beranek and Newman, Inc.
BER	Bit Error Rate
C&C	Command & Control
C&DH	Command and Data Handling
CCL	Comet Command Language
CCR	Configuration Change Request
CCSDS	Consultative Committee on Spacecraft Data Systems
CDE	Common Desktop Environment
CDS	Cell Directory Service
CGI	Common Gateway Interface
CM	Configuration Management
CMD	Command
COE	Center of Expertise
CORBA	Common Object Request Broker Architecture
COSE	Common Operating Software Environment
COST LESS	Code O Success Team Lifecycle Effectiveness for Strategic Success

COTS	Commercial Off The Shelf
CPU	Central Processing Unit
CRC	Cyclical Redundancy Check
CSC	Communications Services Component
CVT	Current Value Table
DAC	Digital to Analog Converter
DAD	Data Distribution component
DBMS	Database Managment System
DCE	Distributed Computing Environment
DCN	Document Change Notice
DEC	Digital Equipment Corporation
Defs	Definitions
DES	Data Encryption Standard
DFS	Distributed File System
DHDS	Digital History Data Store
DNS	Domain Name Service
DoD	Department of Defense
DRA	Data Reduction and Analysis
DSN	Deep Space Network
DTS	Distributed Time Service
EDA	Enterprise Data Access
ELV	Expendable Launch Vehicle
Email	Electronic Mail
EOS	Earth Observing System
ESA	European Space Agency
FDDI	Fiber Distributed Data Interface
FEDS	Front End Data System
FOT	Flight Operations Team
ftp	File Transfer Protocol (application)

FTP	File Transfer Protocol (protocol)
FY95	Fiscal Year 1995
FY96	Fiscal Year 1996
FYxx	Fiscal Year xx
GBytes	Gigabytes
GDS	Ground Data System
Genie	Generic Inferential Executor
GenSAA	Generic Spacecraft Analysis Assistant
GN	Ground Network
GOTS	Government Off The Shelf
GPIB	General Purpose Interface Bus
GPS	Global Positioning System
GREAS	Generic Resource, Event, and Activity Scheduler
GS	Ground Station
GSFC	Goddard Space Flight Center
GUI	Graphical User Interface
HCI	Human Computer Interface
HP	Hewlett-Packard
HPOP	High Precision Orbit Propagator
HST	Hubble Space Telescope
HTML	Hypertext Markup Language
HTTP	HyperText Transport Protocol
I&T	Integration & Test
IBM	International Business Machines, Inc.
ICCCM	Inter-Client Communications Conventions Manual
ICS	Interface & Control Systems, Inc.
ID	Identifier
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineering

IETF	Internet Engineering Task Force
IGSE	Instrument Ground Support Equipment
IMT	Intelligent Monitoring Toolkit
IO	Input Output
IP	Internet Protocol
IPv6	Internet Protocol, Version 6
IPC	Interprocess Communications
ISDN	Integrated Services Data Network
JPEG	Joint Photographic Experts Group
JPL	Jet Propulsion Laboratory
Kbps	Kilobits per second
KKI	Kisak-Kellogg Inc.
KQML	Knowledge Query and Manipulation Language
L&EO	Launch and Early Orbit
LAN	Local Area Network
LEO	Low Earth Orbit
LPC	Language Processing Component
LTIS	Loral Test & Integration Systems
MAGIC	Multimission Advanced Ground Intelligent Control
Mbps	Megabits per second
MCS	Mission Control System (Altair MCS)
MFILE	Message File
MIB	Management Information Base
MIDEX	Medium Explorer
MIME	Multipurpose Internet Mail Extensions
MLS	Multi-level Security
MMS	Manufacturing Message Specification
MO	Mission Operations
MO&DA	Mission Operations and Data Analysis

MO&DSD	Mission Operations and Data Systems Directorate
MOC	Mission Operations Center
MOCA	Mission Operations Control Architecture
MODNET	Mission Operations Directorate Network
MOPS	Mission Operations Planning System
MPEG	Moving Picture Experts Group
MS	Microsoft, Inc.
MTA	Mail Transfer Agent
MTBF	Mean Time Between Failures
MTTR	Mean Time To Restore
NASA	National Aeronautics and Space Administration
NASCOM	NASA Communications
NFS	Network File Services
NHB	NASA Handbook
NIC	Network Interface Card
NIST	National Institute for Standards and Technology
NMS	Network Management System
NT	New Technology
OAC	Orbit Analysis Component
OB	Onboard
ODBC	Open Database Connectivity
OODBMS	Object Oriented Database Management System
ORB	Object Request Broker
OS	Operating System or Open System
OSE	Open System Environment
OSF	Open Systems Foundation
OSI	Open Systems Interconnect
OTS	Off The Shelf
Pacor II	Packet Processor II

PC	Personal Computer
PCI	Peripheral Component Interconnect (bus)
PDC	Programmable Device Controller
PDP	Production Data Processing
PFS	Platform File Services
PGP	Pretty Good Privacy
PI	Principal Investigator
pixmap	Pixel Map
PODS	Precision Orbit Determination System
PPS	Packet Processing System
PTCW	Primary Test Conductor Workstation
RAID	Redundant Array of Inexpensive Disks
RDBMS	Relational Database Management System
REM	Remote Equipment Monitor
Renaissance	Reusable Network Architecture for Interoperable Space Science, Analysis, Navigation, and Control Environment
RF	Radio Frequency
RFC	Request For Comment
RISC	Reduced Instruction Set Computer
RLC	Recording & Logging Component
RMON	Remote Monitoring Protocol
RMP	Reliable Multicast Protocol
ROM	Read Only Memory
RPC	Remote Procedure Call
S/C	Spacecraft
SA	Single Access
SAC	System Administration Component
SBC	Single Board Computer
SCL	Spacecraft Command Language

SCPS	Space Communication Protocol Standards
SFDU	Standard Formatted Data Unit
SMEX	Small Explorer
SMEX-Lite	Small Explorer - (Lower Cost Version)
SMP	Symmetrical Multiprocessor
SMTP	Simple Mail Transfer Protocol
SN	Space Network
SNMP	Simple Network Management Protocol
SNMPc	Simple Network Management Protocol - Personal Computer
SNMPv1	Simple Network Management Protocol Version 1
SNMPv2	Simple Network Management Protocol Version 2
SONET	Synchronous Optical Network
Specs	Specifications
SQL	Standard Query Language
STI	Software Technology, Inc.
STK	Satellite Toolkit
STOL	System Test and Operations Language
SuperMOCA	Space Project Mission Operation Control Architecture
SYM	Symbol Processing component
TBD	To Be Determined
TBR	To Be Resolved
TBS	To Be Supplied
Tcl	Tool Control Language
TCP	Transmission Control Protocol
TCW	Test Conductor Workstation
TDRS	Tracking and Data Relay Satellite
TLI	Transport Level Interface
TLM	Telemetry
TPCE	Telemetry Processing Control Environment

TPOCC	Transportable Payload Operations Control Center
TSDS	Decommutated Sequential Telemetry Stream Server
TSTOL	TPOCC System Test and Operations Language
UDP	User Datagram Protocol
UID	User Interface Description
UIL	User Interface Language
US	United States
UTC	Coordinated Universal Time
UTP	Unshielded Twisted Pair
VC	Virtual Channel
VCID	Virtual Channel Identifier
VCDU	Virtual Channel Data Unit
VME	Versa Module European
VUE	Visual User Environment
WAN	Wide Area Network
WWW	World Wide Web
X11R5	X Window System Release 5
XTE	X-Ray Timing Explorer

Glossary

Application	A piece of software that provides a primary mission support capability, e.g., spacecraft command and control.
Cell	The highest level of mission system component, implemented using the standard local area architecture and technology.
Center of Expertise	A system support entity that provides the mission with expert support for development or operations beyond that which is normally available among the mission operations team. COEs may be used for nominal support or may be called upon for contingency situations.
Client	A software process that exchanges information with a server. In the Renaissance second generation architecture, clients include applications software as well as other servers.
Client-Server	A form of distributed processing in which multiple clients interact with a server process controlling an information resource.
Customer	The person, organization, or group which provides the source of funds for developing of the mission system. Indirectly, the organization or community which is using the space system to address a research problem or provide a commercial service. The funding and using customer may be the same organization.
Dataset	A collection of information that is generated for use at an arbitrarily later time. Data delivered as datasets include software images, data logs, and customer products. Subsets of the entire collection are typically extracted for use.
Dataset Server	A server that handles the distribution of complete or partial datasets from producers to consumers.
Dedicated	Describes a resource that is developed and operated in support of a single mission. A dedicated resource is totally managed by the mission.
Domain	A sphere of activity within the system, within which common services are provided.
Extended Operations	The phase of the mission life cycle, after the mission's primary goals have been met, in which secondary goals are pursued. Extended operations usually begin after some number of months or years after launch.
External	Anything that is not provided as part of a standard cell. External interfaces typically provide access to shared and contingency resources.

	Interface types may include data interfaces, schedule interfaces, operational messages, and management interfaces.
Front End	System that performs protocol conversion between terrestrial communications and space-to-ground communications for telemetry (return link) and commands (forward link). Currently, front ends are comprised of special hardware and software.
Ground Data System	A portion of the mission system, located on the ground, that is responsible for capture and processing of data returned by the spacecraft and for the preparation and uplink of commands and data to the spacecraft.
Ground Station	A system that provides data transport and protocol conversion between terrestrial communications networks and the RF communications necessary to communicate with spacecraft after launch. Ground stations are typically composed of one or more antennae and RF equipment. The definition is expanded to include the ground elements of systems such as the Space Network which use relay satellites to communicate with spacecraft in orbit. Ground stations may include tracking support as a supplemental function.
Integration & Test	The phase of the mission life cycle prior to launch during which the spacecraft components are integrated and checked out, followed by mission system establishment and testing.
Internet	A world-wide, public network used for interchange of data and information.
Internet Protocol	A suite of communications protocols and applications developed for transmission of information across the Internet. Also widely used for transmission on local area networks and private wide area networks. More specifically, the protocol which provides the network layer in the OSI model, covering packet addressing and routing.
Local Area Network	A network of nodes (addresses) within a single domain of lowest level Internet Protocol addresses, usually implemented with technology appropriate to small (~100 m) networks.
Message	A packet of data containing information that must be acted upon promptly. Data delivered as messages include real-time telemetry and commands and operations event data
Message Server	A server that handles the delivery of distinct messages from producers to consumers.
Mission	A coordinated effort to develop and operate a spacecraft (or group of related spacecraft) in support of an overall scientific goal.

Mission Operations	All activities supporting accomplishment of mission goals, including provision of customer data, mission system resource management, and system evolution.
Mission System	The collection of dedicated and shared resources needed to carry out the goals of a mission. The mission system consists of both ground and space resources.
Mission System Management	The operation and control of dedicated resources and scheduling and monitoring of shared resources used to meet the mission's goals.
Multi-Mission	A coordinated effort to develop and/or operate two or more missions using the same resources.
Off-the-Shelf	Hardware or software that can be integrated into a system "as is" or with only configuration modifications. Includes commercially available (Commercial Off-the-Shelf or COTS) and government-developed (Government Off-the-Shelf or GOTS).
Operations Phase	The phase of the mission life cycle, after launch, during which the mission's primary goals are being pursued.
Operations Center	A facility within the mission system that is responsible for some portion of the mission operations, normally implemented as a single cell.
Payload Analysis	Data processing and analysis of payload generated data. Analysis includes technical analysis of data to pursue primary customer goals as well as engineering analysis of data to monitor instrument health and safety.
Peer-To-Peer	A form of distributed processing in which applications communicate directly with each other as equal partners.
Publish/Subscribe	A client-server interaction method in which consumers receive requested types of messages or data in continuing streams, receiving new messages or data updates as the server receives them from producers, or on a periodic basis.
Request/Response	A client-server interaction method in which message consumers determine what messages or data they want by making an individual specific request to the server for the data.
Server	A software process that manages an information resource, e.g., an information repository or the human-computer interface.
Shared	A resource that is developed and operated in support of multiple missions. Examples include multi-mission operations centers, institutional ground stations, and the Internet. Shared resources are managed by an entity external to the mission.

Subdomain	A group of tightly-coupled applications within a specific cell. Subdomains are characterized by a specific, shared data interface which may vary from the design standard client-server model.
Wide Area Network	A network that is used for cell-to-cell communication and cell-to-external communication.